TUGAS AKHIR - KI141502

PENGGUNAAN METODE ANALISIS SPASIAL DAN GENETIC ALGORITHMS (GA) PADA SISTEM PENENTUAN LOKASI OPTIMUM SPBU

MUHAMMAD YARJUNA ROHMAT
NRP 5112100158

Dosen Pembimbing I
Dr.tech. Ir. R. V. Hari Ginardi, M.Sc.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

TUGAS AKHIR - KI141502

PENGGUNAAN METODE ANALISIS SPASIAL DAN GENETIC ALGORITHMS (GA) PADA SISTEM PENENTUAN LOKASI OPTIMUM SPBU

MUHAMMAD YARJUNA ROHMAT
NRP 5112100158

Dosen Pembimbing I
Dr.tech. Ir. R. V. Hari Ginardi, M.Sc.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES - KI141502

AN OPTIMUM GAS STATION LOCATION ALLOCATION USING SPATIAL ANALYSIS AND GENETIC ALGORITHMS (GA)

MUHAMMAD YARJUNA ROHMAT
NRP 5112100158

Supervisor I
Dr.tech. Ir. R. V. Hari Ginardi, M.Sc.

Supervisor II
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENGGUNAAN METODE ANALISIS SPASIAL DAN *GENETIC ALGORITHMS (GA)* PADA SISTEM PENENTUAN LOKASI OPTIMUM SPBU

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
MUHAMMAD YARJUNA ROHMAT
NRP : 5112 100 158

Disetujui oleh Dosen Pembimbing Tugas Akhir

1. Dr.tech. Ir. R. V. Hari Ginardi, M.Sc. .....
   NIP. 19650518 199203 1 003 (Pembimbing 1)

2. Dini Adni Navastara, S.Kom., M.Sc. .....
   NIP. 19851017 201504 2 001 (Pembimbing 2)

SURABAYA
JUNI, 2016

[Halaman ini sengaja dikosongkan]

Penggunaan Metode Analisis Spasial dan Genetic Algorithms (GA) pada Sistem Penentuan Lokasi Optimum SPBU

Nama Mahasiswa	**: MUHAMMAD YARJUNA ROHMAT**
NRP	**: 5112100158**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dr.tech. Ir. R. V. Hari Ginardi, M.Sc.**
Dosen Pembimbing 2	**: Dini Adni Navastara, S.Kom., M.Sc.**

Abstrak

Stasiun Pengisian Bahan Bakar Umum (SPBU) merupakan salah satu fasilitas yang keberadaannya sangat vital bagi masyarakat perkotaan. Karena ketergantungan masyarakat kepada SPBU, SPBU harus berada dekat dengan lingkungan masyarakat. Namun tak dapat dipungkiri bahwa pendirian lokasi SPBU juga perlu untuk mempertimbangkan aspek hukum, lebih spesifiknya aspek Rencana Tata Ruang Wilayah (RTRW). Oleh karena itu, dalam usaha penentuan lokasi SPBU, diperlukan analisis multi aspek yang tepat agar lokasi SPBU tersebut tidak menyalahi RTRW namun tetap memperhatikan sisi ekonomi.

Dalam Tugas akhir ini, penulis membuat sebuah alat bantu berbasis Python (Python Toolbox) pada ArcGIS 10.3 untuk menentukan lokasi optimum SPBU menggunakan metode Genetic Algorithm dengan kriteria jarak terhadap rumah sakit, sekolah, SPBU eksisting dan populasi. Data yang digunakan dalam proses uji coba merupakan data yang mencakup kriteria-kriteria pada kota Surabaya bagian Barat. Berdasarkan hasil implementasi menggunakan Genetic Algorithm dengan parameter jumlah individu sebesar 200 dan interval sebesar 1000, sistem dapat melakukan pencarian alternatif lokasi SPBU dengan nilai konsistensi sebesar 0.92.

Kata Kunci: Analisis Spasial, Genetic Algorithm, SPBU, GIS.

[Halaman ini sengaja dikosongkan]

AN OPTIMUM GAS STATION LOCATION ALLOCATION USING SPATIAL ANALYSIS AND GENETIC ALGORITHMS (GA)

Student's Name	**: MUHAMMAD YARJUNA ROHMAT**
Student's ID	**: 5112100158**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Dr.tech. Ir. R. V. Hari Ginardi, M.Sc.**
Second Advisor	**: Dini Adni Navastara, S.Kom., M.Sc.**

Abstract

Gas Station (SPBU) is one of vital facilities which important for urban communities. Because of the dependence of community to gas stations, gas stations must be located close to the community. However, there is no doubt that the gas station building also need to consider the legal aspects, more specific is Spatial Planning or Rencana Tata Ruang Wilayah (RTRW). Therefore, in determining the location of gas station, is necessary to do the right multiaspects analysis so that the location of gas station does not violate RTRW but still consider the economic side.

In this thesis, author makes a phyton based tools (phyton toolbox) on ArcGIS 10.3 for determining the optimum location of gas stations using Genetic Algorithm method with criteria of distance to hospitals, schools, existing gas station, and population of communities. Data used in the trial process is data that includes criteria in West of Surabaya City. As result, based on the implementation of Genetic Algorithm with parameter individu number of 200 and interval of 1000, Python Toolbox can do a search for an gas station alternative location with the consistency value of 0.92.

Keywords: Spatial Analysis, Genetic Algorithm, Gas Station, GIS

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Puji syukur penulis kehadirat Tuhan YME karena berkat rahmat dan karunia-NYA penulis dapat menyelesaikan Tugas akhir yang berjudul:

Penggunaan Metode Analisis Spasial dan Genetic Algorithms (GA) pada Sistem Penentuan Lokasi Optimum SPBU

Tugas akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis ingin menyampaikan terima kasih yang sebesar-besarnya atas dukungan dan semangat yang diberikan serta membantu penulis baik secara langsung ataupun tidak dalam menyelesaikan Tugas akhir ini. Penulis ingin mengucapkan terima kasih kepada

1. Allah SWT karena berkat rahmat dan karunia-Nya penulis berhasil menyelesaikan Tugas akhir dengan baik.
2. Kedua orang tua, dan keluarga penulis, terima kasih atas doa dan bantuan moral dan material selama penulis belajar di Teknik Informatika ITS.
3. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom., selaku ketua jurusan Teknik Informatika ITS
4. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku Koordinator Tugas akhir di Teknik Informatika ITS.
5. Bapak Dr.tech. Ir. R. V. Hari Ginardi, M.Sc. selaku Dosen Pembimbing I Tugas akhir yang telah memberikan bimbingan dan dukungan selama penulis menyelesaikan Tugas akhir.

6. Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing II Tugas akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas akhir.
7. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan ilmu selama penulis kuliah di Teknik Informatika
8. Seluruh staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis kuliah di Teknik Informatika.
9. Rekan-rekan mahasiswa yang telah bersedia membantu penulis selama pengerjaan Tugas akhir.

Penulis mohon maaf apabila terdapat kekurangan dalam penulisan Tugas akhir ini. Kritik dan saran penulis harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga Tugas akhir ini dapat memberikan manfaat yang sebesar besarnya.

Surabaya, Juni 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan Tugas akhir. Diharapkan dari penjelasan dalam bab ini gambaran Tugas akhir secara umum dapat dipahami.

1.1 Latar Belakang

Stasiun Pengisian Bahan Bakar Umum (SPBU) merupakan salah satu fasilitas yang keberadaannya sangat vital bagi masyarakat perkotaan. Hal tersebut dikarenakan mayoritas masyarakat perkotaan merupakan pengendara kendaraan bermotor dan menyebabkan kebutuhan masyarakat akan kehadiran SPBU semakin meningkat. Karena ketergantungan masyarakat kepada SPBU tersebut, SPBU harus berada dekat dengan lingkungan masyarakat, baik lingkungan tempat tinggal maupun lingkungan perkotaan [1].

Namun tak dapat dipungkiri bahwa pendirian lokasi SPBU juga perlu untuk mempertimbangkan aspek hukum, lebih spesifiknya aspek Rencana Tata Ruang Wilayah (RTRW). Kota Surabaya membatasi pembangunan SPBU oleh peraturan pemerintah mengenai Rencana Tata Ruang Wilayah (RTRW) yang secara jelas dipaparkan di dalam peraturan walikota Surabaya nomor 75 tahun 2014 bahwa SPBU diizinkan dibangun di wilayah perdagangan, jasa, dan industri. Oleh karena itu, dalam usaha penentuan lokasi SPBU, diperlukan analisis multi aspek yang tepat agar lokasi SPBU tersebut tidak menyalahi RTRW namun tetap memperhatikan sisi ekonomi dalam rangka memenuhi faktor kebutuhan masyarakat yang tinggi.

Untuk mengatasi permasalahan ini, penerapan Sistem Informasi Geografis (SIG) merupakan salah satu langkah yang dapat digunakan. SIG sudah sangat umum digunakan dalam

kegiatan tata kota karena berbagai kemampuannya untuk mengolah data spasial. Kegiatan utama dalam mengolah data spasial adalah analisis spasial yang hasilnya merupakan sebuah informasi baru yang akan digunakan sebagai dasar pengambilan keputusan, dalam hal ini yaitu lokasi SPBU. Ada banyak metode yang dapat digunakan untuk mendukung kegiatan analisis spasial, salah satunya adalah metode *Genetic Algorithms* (GA). GA dipilih karena kemampuannya untuk menyelesaikan permasalahan optimasi yang harus memenuhi berbagai macam kriteria.

Terdapat banyak penelitian yang berhasil menggunakan metode serupa untuk menyelesaikan permasalahan optimasi lokasi yang berbeda, contohnya seperti penggunaan GA pada aplikasi penentuan *Well Drilling Location* [2] dan *Optimal Location Search* [3]. Kedua literatur tersebut menjelaskan mengenai bagaimana implementasi GA pada masing-masing kasus penempatan suatu lokasi/fasilitas. Hal tersebut menunjukkan bahwa GA mampu untuk menemukan jawaban untuk permasalahan optimasi penentuan lokasi.

Tujuan dari pengerjaan Tugas akhir ini adalah terbentuknya alat bantu yang dapat memberikan titik-titik rekomendasi lokasi SPBU dengan memenuhi kriteria-kriteria yang sebelumnya telah ditentukan dengan mempertimbangkan faktor ekonomi dan peraturan pemerintah mengenai rencana tata ruang wilayah. Oleh karena itu, pada Tugas akhir ini penulis mengimplementasikan GA pada alat bantu yang akan dibuat sebagai metode untuk menentukan titik-titik rekomendasi lokasi SPBU.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas akhir dapat dipaparkan sebagai berikut.

a) Bagaimana menentukan kriteria-kriteria penentuan lokasi SPBU yang potensial?

b) Bagaimana mengolah data geospasial agar dapat digunakan dalam metode perhitungan?
c) Bagaimana menerapkan GA pada data geospasial untuk menentukan lokasi SPBU yang optimal?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas akhir memiliki beberapa batasan, yakni sebagai berikut.

1. Studi kasus penentuan lokasi SPBU adalah wilayah kota Surabaya Barat.
2. Perangkat lunak yang digunakan adalah ArcGIS 10.3.
3. Implemetasi dilakukan menggunakan bahasa pemrograman Python.
4. Data geospasial yang digunakan berasal dari Dinas Cipta Karya dan Tata Ruang dan Badan Perencanaan Pembangunan Kota Surabaya.
5. Kriteria yang digunakan adalah kriteria populasi, jarak dengan SPBU saat ini, rumah sakit dan sekolah.
6. Diasumsikan lahan peletakan lokasi SPBU baru selalu tersedia.

1.4 Tujuan

Tujuan dari pembuatan Tugas akhir ini adalah membuat sebuah alat bantu berbasis Python (*Python Toolbox*) pada ArcGIS 10.3 untuk menentukan lokasi optimum SPBU menggunakan *Genetic Algorithm*.

1.5 Manfaat

Pengerjaan Tugas akhir ini dilakukan dengan harapan dapat membantu pihak pengembang SPBU untuk menentukan lokasi SPBU yang akan dibangun.

1.6 Metodologi

Metodologi yang dipakai pada pengerjaan Tugas akhir ini adalah sebagai berikut:

1. Penyusunan Proposal Tugas akhir
   Tahap awal yang dilakukan dalam pengerjaan Tugas akhir ini adalah penyusunan proposal Tugas akhir. Di dalam proposal diajukan suatu gagasan pembuatan sistem penentuan lokasi optimum SPBU.

2. Studi Literatur
   Tahap ini merupakan tahap pengumpulan informasi yang diperlukan untuk pengerjaan Tugas akhir sekaligus mempelajarinya. Termasuk diantaranya adalah pengumpulan literatur dari buku referensi, jurnal, dan dokumentasi internet yang meliputi hal-hal berikut:
   - Studi literatur tentang konsep dasar GIS.
   - Studi literatur tentang analisis lokasi dan keruangan.
   - Studi literatur tentang *Genetic Algorithms*.
   - Studi literatur tentang ArcGIS 10.3 dan pengembangan *Python Toolbox*.

3. Implementasi dan Pembuatan Perangkat Lunak
   Pada tahap ini dilakukan impelementasi perangkat lunak sesuai dengan rancangan perangkat lunak yang dibuat.

4. Uji Coba dan Evaluasi
   Pada tahap ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat untuk mengetahui kemampuan algoritma yang dipakai, mengamati kinerja sistem, serta mengidentifikasi kendala yang mungkin timbul. Jika terdapat kendala, kekurangan, atau kesalahan dapat diperbaiki demi kelayakan dan keberhasilan aplikasi sesuai dengan tujuan pembuatan Tugas akhir.

5. Penyusunan Laporan Tugas akhir
   Tahap ini merupakan tahap penyusunan buku sebagai dokumentasi dari pelaksanaan Tugas akhir, yang mencakup

seluruh konsep, implementasi, serta hasil pengujian yang telah dilakukan.

1.7 Sistematika Penulisan

Buku ini disusun dengan sistematika penulisan sebagai berikut:

BAB I Pendahuluan

Menjelaskan latar belakang, permasalahan yang diangkat beserta batasan-batasannya, tujuan dan manfaat yang ingin dicapai dari pembuatan Tugas akhir ini, beserta metodologi yang dipakai.

BAB II Dasar Teori

Memaparkan hasil studi literature berupa rangkuman materi tentang konsep dasar GIS, Teori Analisis Lokasi dan Keruangan, *Genetic Algorithms* serta materi lainnya yang berhubungan denagn Tugas akhir ini.

BAB III Metodologi

Membahas rancangan aplikasi yang akan dibangun. Mulai dari deskripsi umum sistem, batasan sistem, proses-proses yang terjadi, dan desain dataBase yang digunakan oleh sistem. Bab ini menunjukkan implementasi dari tiap rancangan yang telah dibuat, meliputi lingkungan implementasi sistem, implementasi proses, implementasi antar muka dan seterusnya.

BAB IV Uji Coba dan Evaluasi

Membahas mengenai pengujian aplikasi berdasarkan skenario-skenario yang telah ditentukan sebelumnya. Hasil dari tiap uji coba akan dievaluasi dan dianalisa dari segi hasil dan data yang diperoleh.

BAB V Kesimpulan dan Saran

Menyatakan kesimpulan-kesimpulan yang didapat dari proses pembuatan Tugas akhir, beserta saran-saran untuk pengembangan selanjutnya.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Pada bab ini dibahas mengenai teori-teori penunjang dalam melakukan implementasi. Di antaranya meliputi pembahasan mengenai konsep dasar GIS, teori analisis lokasi dan keruangan, dan *Genetic Algorithms*.

2.1 Konsep Dasar GIS

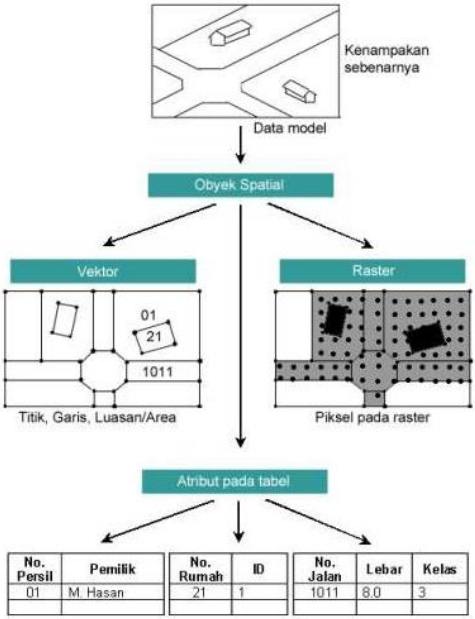
Geographic Information System (GIS) atau Sistem Informasi Geografis (SIG) adalah sistem informasi berbasis komputer yang menyimpan dan mengolah data bereferensi geografis atau data spasial yang dapat dikaitkan dengan data non-spasial (data dalam bentuk tabel), yang dapat digunakan untuk melakukan pemrosesan informasi secara luas termasuk manipulasi, pemodelan, dan analisis [4].

Data spasial adalah data yang berhubungan dengan kondisi geografi misalnya sungai, wilayah administrasi, gedung, jalan raya, dan sebagainya. Data spasial didapatkan dari peta, foto udara, citra satelit, data statistik, dan lain-lain. Sedangkan data non-spasial adalah selain data spasial, yaitu data yang berupa teks atau angka, biasanya disebut dengan atribut.

Data non-spasial ini akan menerangkan data spasial atau sebagai dasar untuk menggambarkan data spasial. Dari data nonspasial ini nantinya dapat dibentuk data spasial.

2.1.1 Representasi Data Geospasial Dalam GIS

SIG merepresentasikan data geografis dengan data spasial yang terbagi ke dalam dua model data, yaitu data raster dan data vektor [5] seperti ditunjukkan pada Gambar 2.1.



Gambar 2.1 Representasi Data GIS

2.1.1.1 Data Raster

Data raster (atau disebut jugan dengan sel grid) adalah data yang dihasilkan dari sistem penginderaan jauh. Pada data raster, obyek geografis direpresentasikan sebagai struktur sel grid yang disebut dengan *pixel* (*picture element*). Pada data raster, resolusi tergantung pada ukuran *pixel*-nya.

Dengan kata lain, resolusi *pixel* menggambarkan ukuran sebenarnya di permukaan bumi yang diwakili oleh setiap *pixel* pada citra. Semakin kecil ukuran permukaan bumi yang direpresentasikan oleh satu sel, semakin tinggi resolusinya, data raster sangat baik untuk merepresentasikan batas-batas yang berubah secara gradual, seperti jenis tanah, kelembaban tanah, vegetasi, suhu tanah, dan sebagainya.

2.1.1.2 Data Vektor

Dalam data vektor, bumi direpresentasikan sebagai suatu bentuk yang terdiri atas garis (*arc/line*), polygon (daerah yang dibatasi oleh garis yang berawal dan berakhir pada titik yang sama), titik/*point* (*node* yang mempunyai label), dan *nodes* (merupakan titik perpotongan antara dua buah garis).

Model data vektor merupakan model data yang paling banyak digunakan, model ini berbasiskan pada titik (*points*) dengan nilai koordinat (x,y) untuk membangun obyek spasialnya. Obyek yang dibangun terbagi menjadi tiga bagian lagi yaitu berupa titik (*point*), garis (*polyline*), dan area (*polygon*).

2.1.2 Analisis Spasial

Analisis spasial adalah suatu teknik atau proses yang melibatkan sejumlah hitungan dan evaluasi logika yang dilakukan dalam rangka mencari atau menemukan potensi hubungan atau pola-pola yang mungkin terdapat di antara unsur-unsur geografis yang terkandung di dalam data digital dengan batas-batas wilayah studi tertentu.

Sebagai sebuah metode, analisis spasial berusaha untuk membantu perencana dalam menganalisis kondisi permasalahan berdasarkan data dari wilayah yang menjadi sasaran. Dan konsep-konsep yang paling mendasari sebuah analisis spasial adalah jarak, arah, dan hubungan. Kombinasi dari ketiganya mengenai suatu wilayah akan bervariasi sehingga membentuk perbedaan yang signifikan yang membedakan satu lokasi dengan yang lainnya. Dengan demikian jarak, arah, dan hubungan antara lokasi suatu objek dalam suatu wilayah dengan objek di wilayah yang lain akan memiliki perbedaan yang jelas. Dan ketiga hal tersebut merupakan hal yang selalu ada dalam sebuah analisis sapasial dengan tahapan-tahapan tertentu tergantung dari sudut pandang perencana dalam memandang sebuah permasalahan analisis spasial [6].

Analisa spasial merupakan sekumpulan metoda untuk menemukan dan menggambarkan tingkatan/ pola dari sebuah fenomena spasial, sehingga dapat dimengerti dengan lebih baik. Dengan melakukan analisis spasial, diharapkan muncul infomasi baru yang dapat digunakan sebagai dasar pengambilan keputusan di bidang yang dikaji. Metoda yang digunakan sangat bervariasi, mulai observasi visual sampai ke pemanfaatan matematika/statistik terapan [7].

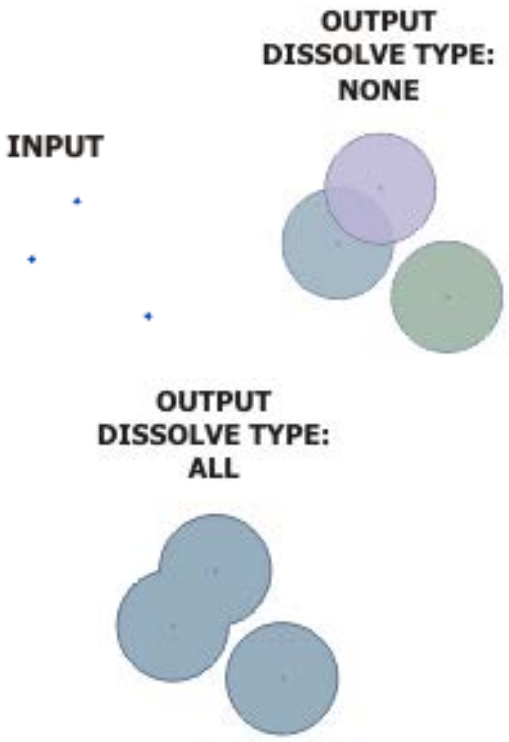
2.2 ArcGIS dan Python Toolbox

ArcGIS adalah salah satu perangkat lunak yang dikembangkan oleh ESRI (*Environment Science & Research Institute*) yang merupakan kompilasi fungsi-fungsi dari berbagai macam perangkat lunak GIS yang berbeda seperti GIS desktop, server, dan GIS berbasis web. *Software* ini mulai dirilis oleh ESRI pada tahun 2000. Produk utama dari ArcGIS adalah ArcGIS desktop, dimana ArcGIS desktop merupakan perangkat lunak GIS profesional yang komprehensif dan dikelompokkan atas tiga komponen yaitu: ArcView (komponen yang fokus ke penggunaan data yang komprehensif, pemetaan dan analisis), ArcEditor (lebih fokus ke arah pengolahan data spasial) dan ArcInfo (lebih lengkap dalam menyajikan fungsi-fungsi GIS termasuk untuk keperluan analisis *geoprocessing*) [8].

Berikut ini adalah penjelasan *tools* dan proses-proses pada ArcGIS yang digunakan untuk analisis spasial pada Tugas akhir ini:

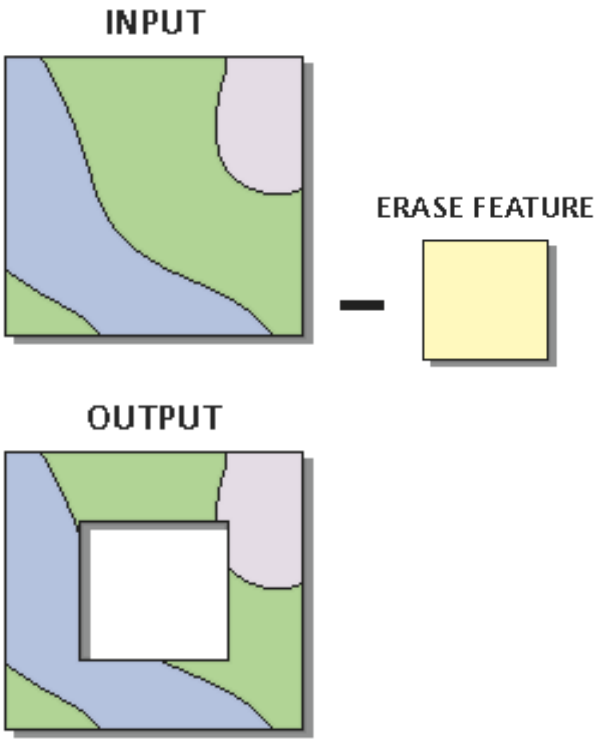
1. *Buffer*

   Tool ini berada pada *toolbox Analysis tools* → *Proximity* → *Buffer*. *Tool* ini berguna untuk membuat *buffer* berbentuk *polygon* dari suatu *input feature* dengan jarak yang spesifik. Pada *tool* ini, *output* dapat dibuat di-*disolve* atau tidak. *Output* yang di-*disolve* maka *output* nya akan menjadi satu *polygon* seperti diilustrasikan pada Gambar 2.2.



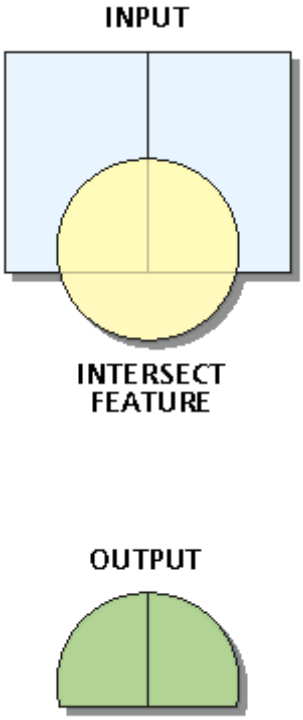
Gambar 2.2 Ilustrasi Fungsi *Buffer*

2. *Erase*
   Tool ini berada pada *toolbox Analysis tools* → *Overlay* → *Erase*. Seperti diilustrasikan pada Gambar 2.3, *tool* ini berguna untuk membuat sebuah *feature* baru dengan cara meng-*overlay*-kan nya dengan *polygon feature erase*. Kemudian dari *feature* awal yang akan tersisa hanya yang diluar dari *polygon feature erase*.



Gambar 2.3 Ilustrasi Fungsi *Erase*

3. *Intersect*
   Tool ini berada pada *toolbox Analysis tools* → *Overlay* → *Intersect*. *Tool* ini berguna untuk menghitung persimpangan geometris dari suatu *feature input*. *Feature* atau bagian dari *feature* yang *overlay* satu sama lain akan masuk dalam bagian *feature output* dan tabel atribut *feature output* akan berisi dari tabel atribut *feature-feature* yang saling *overlay*. Ilustrasinya dapat dilihat pada Gambar 2.4.

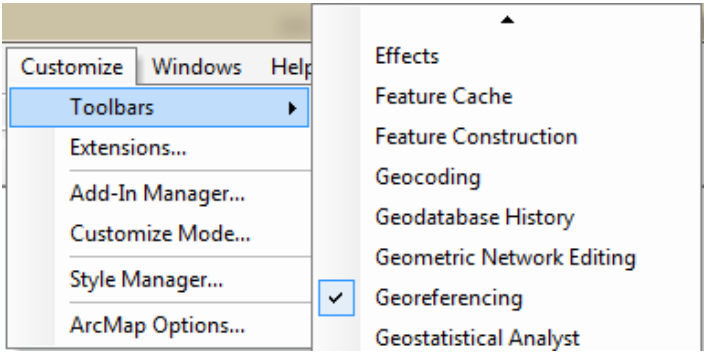


Gambar 2.4 Ilustrasi Fungsi *Intersect*

4. *Georeferencing*

Untuk merepresentasikan permukaan bumi yang berbentuk bola (mendekati bola/elipsoid) ke dalam bentuk peta (gambar 2 dimensi), diperlukan sebuah model. Model ini dikenal sebagai sistem koordinat. Penggunaan sistem koordinat merupakan ciri khas utama GIS karena sistem koordinat inilah yang menunjukkan referensi geografis pada data-data GIS. Dengan kata lain, sistem koordinat berguna dalam mendefinisikan posisi objek di atas permukaan bumi. Pada umumnya, terdapat dua jenis sistem

koordinat yakni Sistem Koordinat Geografis (*Geographic Coordinate System*), dan *Projected Coordinate System* yang lazimnya digunakan di Indonesia seperti UTM (*Universal Transverse Mercator*). Untuk mendefinisikan system koordinat pada berbagai macam data spasial, ArcMap memfasilitasi kita dengan *tools georeferencing*. Dapat di akses pada *Customize -> Toolbars -> Georeferencing* (pastikan *toolbar georeferencing* tercentang) seperti terlihat pada Gambar 2.5 dan Gambar 2.6.



Gambar 2.5 *Georeferencing Tools*

Gambar 2.6 *Georeferencing Toolbar*

5. Digitasi
   Digitasi adalah proses *tracing* di atas data raster (citra atau peta analog) yang akan menghasilkan data vektor (*point*, *polyline*, atau *polygon*). Pada umunya, data raster yang digunakan/di-digitasi adalah hasil proses *georeferencing* atau peta yang telah memiliki sistem koordinat dan terletak pada koordinat yang sesuai. Dengan melakukan digitasi, data vektor yang dihasilkan dapat diimbuhi dengan data

atribut selain dari data spasialnya. Untuk menghasilkan data vektor ini, dibutuhkan media/*file* untuk diisi atau di-*update* dengan data vektor yang baru. Untuk itu dibutuhkan *file* vektor berformat standar ArcGIS yaitu *shapefile* ataupun *geodatabase file*.

Dalam keperluan analisis spasial (*geoprocessing*), ArcGIS menyediakan beberapa *tools* yang dapat digunakan secara langsung oleh pengguna seperti *buffer analysis tools*, *erase analysis tools*, *intersect analysis tools*, dan lain sebagainya. ArcGIS juga memiliki kemampuan agar pengguna dapat membangun *tools* sendiri untuk permasalahan yang lebih beragam.

Tools yang dapat dibuat sendiri oleh pengguna ini dinamakan *Python Toolbox*. *Python Toolbox* adalah *geoprocessing toolbox* yang diciptakan seluruhnya menggunakan bahasa pemrograman Python. *Python Toolbox* dan *tools* yang berada di dalamnya tampak, bertingkah, dan bekerja persis seperti *toolbox* yang sudah disediakan dalam ArcGIS.

Berikut merupakan beberapa keuntungan dari *Python Toolbox*:

1. Memungkinkan pengguna untuk mengambil keuntungan dari pengetahuan Python dan secara cepat membuat prototipe dan menciptakan alat *geoprocessing* yang berfungsi penuh.
2. *Tools* yang dibuat adalah bagian secara keseluruhan dari geoprocessing, seperti *tools* lain yang dapat dibuka dari kolom pencarian atau katalog, menggunakannya dalam *Modelbuilder* dan Python *window*, dan memanggilnya di dalam *script*.

2.3 Teori Analisis Lokasi dan Keruangan

Analisis lokasi dan keruangan memegang peranan penting dalam bidang perencanaan wilayah terutama dalam permasalahan penentuan sebuah lokasi. Analisis lokasi adalah

analisis yang dilakukan pada permasalahan lokasi yang berkembang, dan terutama pada kebanyak kasus terkait pada suatu wilayah yang mempertimbangkan batas-batas geografis, politik, topografi, ekonomi, ekologi, dan populasi [9].

Faktor – faktor penentu lokasi dapat dibagi menjadi 4 bagian [10] yaitu sebagai berikut:

- Faktor Teknologi
- Faktor Ekonomi dan Geografi
- Faktor Politis
- Faktor Sosial

Dalam referensi lain [11] , dikatakan bahwa dalam permasalahan retail, untuk membuka gerai baru, daftar berikut ini dapat dimanfaatkan untuk mengetahui lokasi potensial yang tersedia:

- Besarnya populasi dan karateristiknya: Jumlah penduduk dan kepadatan pada suatu wilayah menjadi faktor dalam mempertimbangkan suatu area perdagangan ritel.
- Kedekatan dengan sumber pemasok: Pemasok mempunyai pengaruh pada peritel dalam hal kecepatan penyediaan, kualitas produk yang terjaga, biaya pengiriman, dan lain-lain.
- Basis ekonomi: Industri daerah setempat, potensi pertumbuhan, fluktuasi karena faktor musiman, dan fasilitas keuangan di daerah sekitar yang harus diperhatikan peritel.
- Ketersediaan tenaga kerja: Tenaga kerja yang perlu diperhatikan adalah pada semua tingkat, yaitu tingkat administrative dan lapangan hingga manajemen trainee dan manajerial.
- Situasi persaingan: Penting mengenali jumlah dan ukuran pada peritel di suatu wilayah.
- Fasilitas promosi: Adanya media massa seperti surat kabar dan radio akanmemfasilitasi kegiatan promosi peritel. Ketersediaan lokasi toko: Faktor bagi suatu area perdagangan dan hal-hal yang terkait dengan lokasi adalah:

jumlah lokasi dan jenisnya, akses pada masing-masing lokasi, peluang kepemilikan, pembatasan zona, perdagangan,dan biaya-biaya terkait.

- Hukum dan peraturan: Hukum dan peraturan perlu di perhatikan khususnya jika terdapat perda atau peraturan daerah yang tidak terdapat di daerah lain.

2.4 Normalisasi *Min Max*

Pada normalisasi *min max*, skala rentang yang umum digunakan yaitu 0 hingga 1. Rumus umum skala adalah sebagai dapat dilihat pada Persamaan 2.1.

$$Y_{new} = \frac{(Y - Y_{min})(Y_{newmax} - Y_{newmin})}{Y_{max} - Y_{min}} + Y_{newmin} \quad \textbf{(2.1)}$$

Variabel Y adalah nilai yang akan dinormalisasi. Variabel Y_{min} dan Y_{max} adalah nilai minimal dan maximum pada nilai-nilai atribut dimana Y berada. Variabel Y_{newmin} dan Y_{newmax} adalah nilai minimal dan maximum yang diinginkan. Apabila rentang yang digunakan adalah 0 hingga 1 maka rumus di atas menjadi Persamaan 2.2.

$$Y_{new} = \frac{(Y - Y_{min})}{Y_{max} - Y_{min}} \quad \textbf{(2.2)}$$

2.5 Genetic Algorithms (GA)

Genetic Algorithm adalah algoritma komputasi yang diinspirasi teori evolusi yang kemudian diadopsi menjadi algoritma komputasi untuk mencari solusi suatu permasalahan dengan cara yang lebih alamiah, *Genetic Algorithm* juga merupakan algoritma pencarian secara heuristik. Salah satu fungsinya ialah untuk mencari solusi atas permasalahan optimasi kombinasi, yaitu mendapatkan suatu nilai solusi optimal terhadap suatu permasalahan yang mempunyai banyak kemungkinan solusi. Teori dasar dari *Genetic Algorithm*

dikembangkan oleh John Holland awal tahun 1975 di Universitas Michigan, Amerika Serikat. Dimana prinsip *Genetic Algorithm* diambil dari teori Darwin yaitu setiap makhluk hidup akan menurunkan satu atau beberapa karakter ke anak atau keturunannya.

Suatu *Genetic Algorithms* standar membutuhkan dua hal untuk didefinisikan [12], yaitu:

1. Sebuah *genetic representation* dari sebuah *solution domain* (domain solusi),
2. Sebuah *fitness function* untuk mengevaluasi sebuah domain solusi.

Kedua hal tersebut kemudian akan melalui 4 proses utama yaitu sebagai berikut:

1. Inisialisasi

   Pada awalnya solusi individual akan secara acak dibuat dalam bentuk sebuah inisial populasi. Besar populasinya sangat tergantung pada keadaan masalah itu sendiri, tapi biasanya populasi mengandung sekitar beberapa ratus atau bahkan ribuan solusi yang mungkin. Secara sederhana, populasinya dibuat secara acak, dengan mengcover seluruh kemungkinkan solusi (*search space*). Cara lainnya, solusinya mungkin bisa di *seeded* pada area dimana kemungkinan besar ditemukan solusi optimalnya.
2. Seleksi

   Seiring dengan berjalannya algoritma, suatu bagian pada populasi akan dipilih (*selected*) untuk membuat suatu generasi baru. Solusi individual tersebut dipilih melalui suatu *fitness-based process*, dimana solusi pencocok (*fitter*, yang diukur oleh suatu *fitness function*) akan menyatakan kemungkinan terpilih. Beberapa metode seleksi menggunakan nilai kecocokan tersebut dan kemudian memilih solusi terbaik dari situ. Metode lain hanya menggunakan solusi acak dari populasi, sehingga proses ini mungkin akan memakan waktu sangat lama. Sebagian bear fungsi bersifat *stochastic* dan didesain agar

sebagian kecil dari solusi yang baik terpilih. Hal ini menolong untuk menjaga keanekaragaman pada suatu populasi cukup besar, mengurangi terjadinya prematur konvergen solusi pada populasi.

3. Reproduksi

   Langkah selanjutnya adalah dengan membuat generasi kedua dari populasi yang ada melalui *genetic operator: crossover* (persilangan), dan atau *mutation* (mutasi). Untuk setiap solusi baru yang dibentuk, sebuah pasangan *parent* atau orang tua solusi dipilih dari kumpulan populasi sebelumnya. Dengan membuat sebuah *child* atau anak solusi menggunakan metoda diatas, yaitu persilangan dan mutasi, sebuah solusi baru telah dibuat, dimana pada umumnya akan memwarisi bagian-bagian dari orang tuanya. Orang tua baru dipilih lagi dan membuat suatu anak solusi lagi, dan berlanjut sampai suatu populasi solusi baru dengan ukuran yang cukup terbentuk. Proses ini akan menghasilkan suatu generasi baru dimana kromosomnya berbeda dengan generasi sebelumnya. Secara umum rata-rata nilai kecocokannya (*fitness*) akan meningkat melalui prosedur ini, karena hanya organisme-organisme terbaik yang dipilih dalam pembentukan populasi selanjutnya, bersama dengan beberapa yang agak cocok dengan solusinya, alasannya sudah disebutkan di atas.

4. Terminasi

   Proses tersebut diatas akan terus dilakukan sampai suatu kondisi terminasi/berhenti ditemukan. Kondisi terminasi/berhenti yang umum dipergunakan yaitu:

   - Suatu solusi ditemukan yang memenuhi kriteria minimal.
   - Generasi telah mencapai suatu tingkat tertentu.
   - *Budget* yang dialokasikan (misalnya waktu komputasi) telah dicapai.
   - Solusi dengan nilai kecocokan tertinggi akan mencapai atau telah mencapai suatu batas dimana

proses selanjutnya yang akan dilakukan tidak akan menghasilkan hasil yang lebih baik.

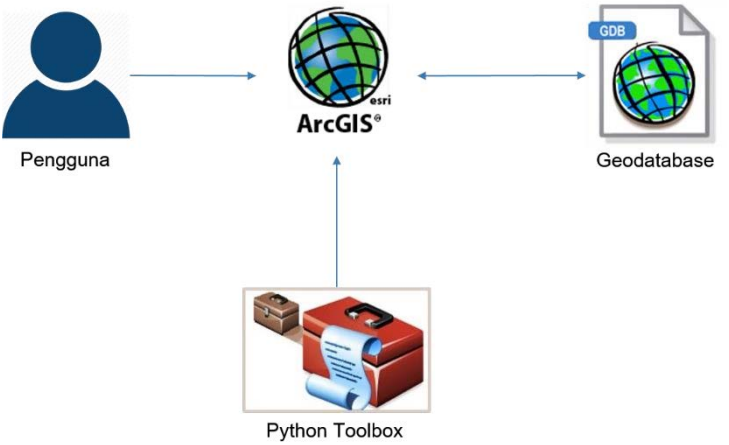
- Inspeksi secara manual dan berkala.
- Kombinasi dari berbagai macam cara terminasi di atas.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

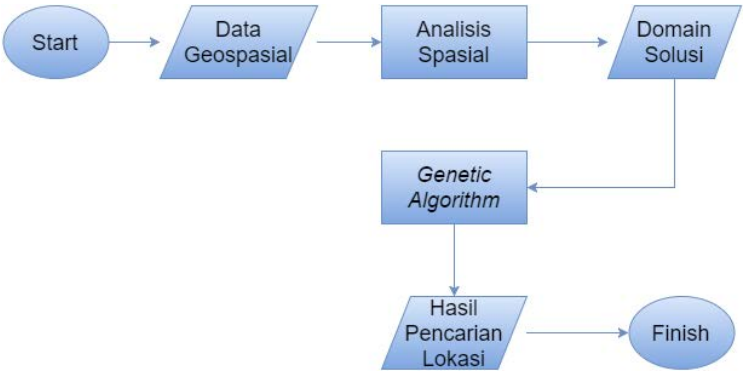
Pada bab ini dijelaskan mengenai rancangan sistem perangkat lunak yang akan dibuat. Perancangan yang dijelaskan meliputi data dan proses. Data yang dimaksud adalah data yang akan diolah dalam perangkat lunak sehingga tujuan Tugas akhir ini bisa tercapai. Proses yaitu tahap-tahap yang ada dalam sistem sebagai pengolah data meliputi Analisis Spasial dan pencarian lokasi optimal menggunakan *Genetic Algorithm*.

3.1 Deskripsi Umum Perangkat Lunak

Perangkat lunak yang akan dibuat pada Tugas akhir ini merupakan sebuah alat bantu (*toolbox*) berbasis Python (*Python Toolbox*) pada ArcGIS 10.3 yang digunakan untuk menentukan lokasi optimum SPBU. Gambar 3.1 menunjukkan arsitektur sistem dan Gambar 3.2 menunjukkan diagram alir dari rancangan *Python Toolbox* yang akan dibangun.



Gambar 3.1 Arsitektur Sistem



Gambar 3.2 Diagram Alir

Berdasarkan Gambar 3.2 dapat diketahui bahwa Tugas akhir ini akan menghasilkan sebuah *Python Toolbox* pada ArcGIS 10.3 yang akan memberikan lokasi alternatif SPBU menggunakan *Genetic Algorithm* berdasarkan kriteria pada data-data masukan.

3.2 Data

Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan.

3.2.1 Data Masukan

Data masukan adalah data yang digunakan sebagai masukan dari *Python Toolbox* yang dibangun. Data yang digunakan adalah data berekstensi *shapefile* (.shp) yang meliputi data-data berikut: jalan kota Surabaya bagian barat, kepadatan penduduk per-kecamatan kota Surabaya bagian barat, sebaran SPBU, sekolah, dan rumah sakit kota Surabaya bagian barat. Data-data tersebut akan menjadi parameter masukan pencarian lokasi alternatif SPBU menggunakan *Genetic Algorithm*.

3.2.2 Data Keluaran

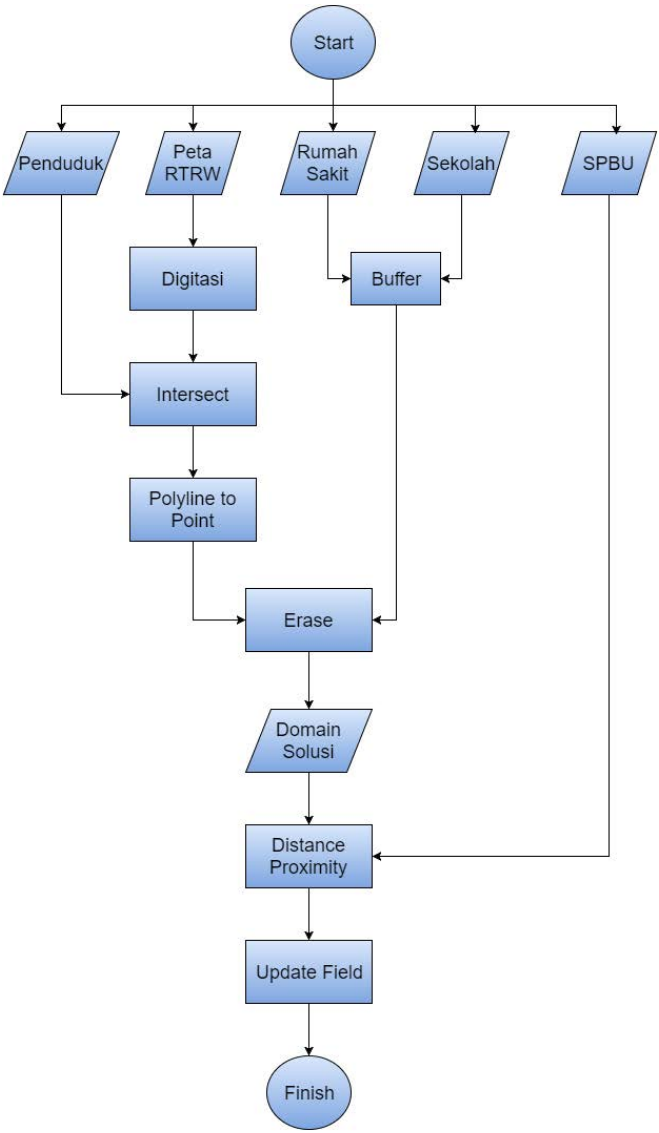
Data masukan akan diolah terlebih dahulu sehingga menghasilkan data yang siap diolah di dalam proses pencarian lokasi optimal sehingga menemukan lokasi optimal. Pengolahan data masukan meliputi hal-hal berikut: Analisis Spasial dan pencarian lokasi optimal menggunakan GA yang akan dibahas lebih lanjut pada sub bab berikutnya. Selanjutnya setelah melalui proses-proses tersebut, sistem akan menghasilkan keluaran berupa lokasi alternatif SPBU dalam bentuk data geospasial.

3.2.3 Pengumpulan Data

Kebutuhan data masukan yang disebutkan pada bab sebelumnya berupa data berekstensi *shapefile* (.shp) yang meliputi data-data berikut: kepadatan penduduk per-kecamatan kota Surabaya bagian barat, sebaran SPBU, sekolah, dan rumah sakit kota Surabaya bagian barat didapatkan dari Dinas Cipta Karya dan Tata Ruang Kota Surabaya. Sedangkan data jalan yang akan menjadi parameter utama pencarian lokasi SPBU didapatkan dari hasil digitasi yang berdasarkan kepada peta Rencana Tata Ruang Wilayah (RTRW) kota Surabaya yang dikeluarkan pada tahun 2014 yang ditunjukkan dalam gambar. Tahap-tahap pengerjaan digitasi peta RTRW menjadi peta jalan akan dijelaskan pada sub bab berikutnya.

3.3 Analisis Spasial

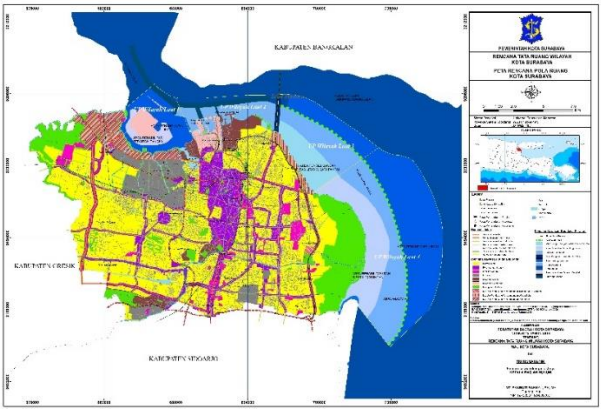
Pada sub bab ini akan dijelaskan mengenai analisis spasial dari permasalahan penentuan lokasi SPBU. Proses-proses yang meliputi hal ini dapat dibagi sebagai berikut: digitasi peta, konversi *polyline to point*, *buffering*, *erasing*, *intersect*, *distance analysis*, dan *update field* seperti digambarkan pada Gambar 3.3. Penjelasan mengenai analisis spasial akan dipaparkan pada sub bab selanjutnya.



Gambar 3.3 Diagram Alir Analisis Spasial

3.3.1 Digitasi Peta

Berdasarkan peraturan walikota Surabaya nomor 75 tahun 2014, rencana pembangunan SPBU yang diizinkan mutlak adalah pembangunan SPBU yang berada di wilayah perdagangan dan jasa serta wilayah industri. Untuk mengetahui pembagian guna lahan tersebut, penulis mengacu pada peta Rencana Tata Ruang Wilayah (RTRW) kota Surabaya yang dikeluarkan pada tahun 2014 yang ditunjukkan dalam Gambar 3.4.



Gambar 3.4 RTRW Kota Surabaya 2012-2034

Digitasi peta dilakukan terhadap data RTRW kota Surabaya yang dijadikan acuan aspek hukum sebagai salah satu kriteria pembatasan lokasi SPBU. Proses ini bertujuan mendapatkan data jalan yang berjenis geometri *polyline* yang akan dijadikan salah satu parameter penentuan lokasi SPBU.

3.3.2 Konversi Polyline to Point

Tahap ini merupakan tahap pembentukan awal domain solusi dari permasalahan penentuan lokasi SPBU. Karena SPBU selalu berada di sekitar jalan, konversi dilakukan dari

data peta jalan yang secara geometri berbentuk *polyline* ke data domain solusi yang secara geometri berbentuk *points*. Konversi dilakukan dengan cara membuat titik di sepanjang jalan dengan interval tertentu.

3.3.3 Buffering

Pada tahap ini dilakukan *buffering* pada *point* rumah sakit dan sekolah untuk menentukan daerah yang berada disekitar rumah sakit dan sekolah. Radius yang ditentukan untuk daerah disekitar *point* tersebut adalah sebesar 150 meter [10].

3.3.4 Erasing

Erasing dilakukan pada domain solusi dengan hasil *buffer point* pada tahap sebelumnya. Artinya, keluaran dari tahap ini adalah domain solusi yang baru yang telah menghilangkan domain solusi yang termasuk pada irisan domain solusi masukan dengan *buffer point*. Domain solusi inilah yang akan digunakan sebagai masukan pada tahap pencarian lokasi optimal menggunakan GA.

3.3.5 Distance Proximity Analysis

Pada tahap ini, domain solusi yang telah didapatkan pada tahap sebelumnya dihitung nilai jaraknya terhadap kriteria yang lain, yaitu sebaran SPBU eksisting. Perhitungan jarak dilakukan dengan menggunakan *Euclidean Distance* untuk setiap kombinasi *point* pada domain solusi dengan setiap *point* pada *layer* sebaran SPBU saat ini. Keluaran pada tahap ini merupakan tabel yang berisi *id* setiap *point* pada kedua *layer* yang dihitung jaraknya.

3.3.6 Pembaruan Atribut

Pembaruan atribut dilakukan untuk memberikan atribut baru kepada data domain solusi. Adapun atribut yang akan diperbaharui adalah atribut jarak dan atribut populasi.

3.3.6.1 Pembaruan Atribut Jarak

Pembaruan atribut jarak dilakukan pada setiap *point* pada domain solusi. Jarak yang diperbaharui pada setiap *point* pada domain solusi merupakan jarak terdekat *point* tersebut untuk setiap kombinasi *point* pada domain solusi dengan setiap *point* pada sebaran SPBU saat ini.

3.3.6.2 Pembaruan Atribut Populasi

Atribut populasi merupakan irisan dari domain solusi dengan peta populasi perkecamatan sehingga domain solusi akan memiliki atribut populasi sesuai dengan jumlah populasi kecamatan irisannya.

3.4 Genetic Algorithms

Pada sub bab ini akan dijelaskan mengenai rancangan implementasi GA terhadap permasalahan penentuan lokasi SPBU.

3.4.1 Representasi Data

Dalam kasus ini, individu direpresentasikan sebagai kumpulan dari *n* buah integer, dimana setiap integer berada dalam *range 1..m* dimana *n* adalah jumlah *point* dan *m* adalah jumlah total *point* pada domain solusi. Setiap nilai integer merepresentasikan *id* setiap *point*.

Sementara itu, nilai atribut setiap *point* direpresentasikan di dalam struktur data yang lain. Contohnya jika terdapat *i* atribut untuk setiap *point* dan *j* total *point*, maka nilai atribut setiap *point* direpresentasikan ke dalam sebuah matriks berukuran *j* x *i*, seperti diilustrasikan pada Gambar 3.5.

n	0	1	2	3	4	5
point id	**4**	**8**	**3**	**1**	**7**	**2**

Gambar 3.5 Representasi Data Individu

3.4.2 Initial Population

Tahap pertama dari GA adalah pembangkitan populasi awal. Dalam Tugas akhir ini, populasi awal dibangkitkan dengan cara melakukan *n* kali *random* bilangan integer dengan *range* antara 1 hingga total jumlah *points* hasil konversi *polyline to points* dan pemanggilan bilangan random ini dilakukan sebanyak *m* kali, dimana *n* adalah jumlah kromosom dan *m* adalah jumlah populasi.

3.4.3 Fitness Evaluation

Untuk menghitung evaluasi *fitness*, digunakan metode normalisasi data menggunakan metode normalisasi *minmax* dan *weighted sum*. Normalisasi *minmax* digunakan untuk menormalisasi data yang didapat dari *geodatabase*. Untuk menghitung *weighted sum*, fungsi objektif yang digunakan adalah jumlah total jarak dengan SPBU terdekat, jumlah total populasi dalam satu individu, dan jumlah total jarak antar titik dalam satu individu sebagaimana ditulis dalam Persamaan 3.1, dimana *a* adalah individu, *w* adalah nilai bobot kriteria, x_{ai} dan y_{ai} adalah nilai penduduk dan jarak dengan SPBU terdekat pada individu *a* ke-*i*, dan *D* adalah fungsi *Euclidean Distance*.

$$f(a) = \sum_{i=o}^{n-1} wx_{ai} + \sum_{i=0}^{n-1} wy_{ai} + \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} D(a_i, a_j) \quad \textbf{(3.1)}$$

Pada contoh nyata, jika diberikan individu seperti pada Gambar 3.5, maka nilai *fitness* individu tersebut adalah jumlah penduduk masing-masing *point id* = 4, 8 ,3, 1, 7, 2 ditambahkan dengan jumlah jarak masing-masing *point id* = 4, 8 ,3, 1, 7, 2 dengan SPBU terdekat ditambahkkan dengan jumlah jarak antar tiap *point id* yaitu kombinasi 2 *point id* pada individu.

3.4.4 Selection

Dalam GA, pembiakan generasi baru dicapai dengan memilih sepasang individu (*parents*) berdasarkan nilai *fitness*, menyilangkan gen anatara kedua individu yang terpilih yang kemudian akan menghasilkan individu baru. Dalam Tugas akhir ini, pemilihan *parents* dilakukan dengan metode *Rank Selection*.

Metode *Rank Selection* adalah metode seleksi dengan mengurutkan individu pada suatu populasi berdasarkan nilai *fitness* mulai dari yang terbaik hingga terburuk, dalam kasus ini *fitness* terbaik merupakan *fitness* dengan nilai terbesar. Kemudian diberikan sebuah nilai probabilitas *p* yang akan menentukan individu mana yang akan dipilih. Semakin besar nilai probabilitas *p*, maka semakin besar probabilitas individu terbaik akan terpilih.

3.4.5 Crossover

Setelah *parents* terpilih, maka tahap yang dilakukan selanjutnya adalah *crossover*, yaitu menyilangkan setiap nilai dari individu antara *parent* 1 dengan *parent* yang lainnya. Metode yang digunakan untuk *crossover* adalah *Uniform Crossover*.

Uniform Crossover mengunakan sebuah *set* yang biasa disebut *mask* sebagai acuan *crossover*. *Set* biasanya berupa angka biner 0 dan 1. Jika pada *index* tertentu nilai *mask* adalah 0, maka tidak terjadi tukar silang antar *parent*. Jika pada *index* tertentu nilai *mask* adalah 1, maka terjadi tukar silang antar

parent pada *index* tersebut seperti ditunjukkan pada Gambar 3.6.

Parent1	4	8	3	1	7	2
Parent2	5	9	6	13	11	10
Mask	1	1	0	1	0	1
Child1	5	9	3	13	7	10
Child2	4	8	6	1	11	2

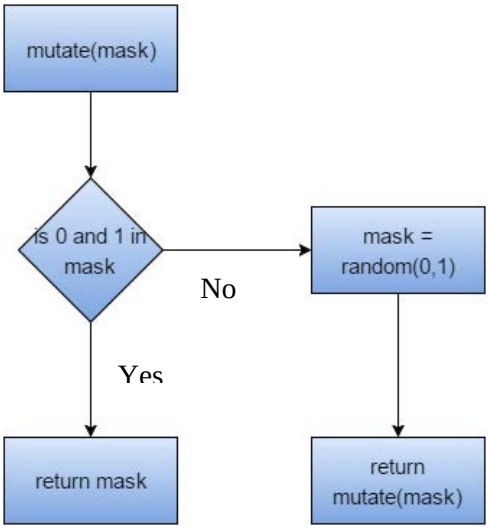
Gambar 3.6 Ilustrasi *Uniform Crossover*

Metode *Uniform Crossover* dipilih karena permasalahan yang dihadapi pada Tugas akhir ini adalah permasalahan kombinasi dan *Uniform Crossover* melakukan *crossover* berdasarkan *item* pada individu sehingga lebih menghasilkan variasi kombinasi individu baru [13] sedangkan metode *crossover* lain rata-rata melakukan *crossover* berdasarkan *segment* pada individu. Penggunaan metode *crossover* lain seperti *single point crossover* atau *twopoint crossover* lebih cocok dugunakan pada permasalahan permutasi karena pada permasalahan tersebut urutan *item* mempengaruhi nilai *fitness*.

3.4.6 Mutation

Dalam *Uniform Crossover*, pembangkitan nilai *random* dari *mask* mempunyai kemungkinan bahwa *mask* memiliki nilai yang seluruhnya setiap elemen sama sehingga menyebabkan tidak adanya perubahan dari hasil *crossover* antar *parents*. Untuk mengatasi hal itu, proses mutasi dilakukan pada pembangkitan *mask* sehingga pada akhirnya *mask* tidak akan memiliki nilai yang seluruhnya sama.

Mutasi pada *mask* dilakukan dengan merancang sebuah fungsi rekursif yang akan mengecek nilai *mask* apakah nilai 0 atau 1 terdapat pada *mask* atau tidak, jika tidak maka akan dilakukan pembentukan nilai *mask* yang baru sehingga terdapat nilai 0 dan 1 pada *mask* tersebut. Rancangan fungsi mutasi rekursif ditunjukkan pada Gambar 3.7.



Gambar 3.7 Rancangan Fungsi Mutasi

3.4.7 Termination

Pada Tugas akhir ini, terminasi dilakukan ketika nilai *fitness* maksimum pada suatu populasi sama dengan nilai *fitness* minimim pada populasi tersebut. Terminasi dilakukan ketika kondisi pada Persamaan 3.2 terpenuhi dimana *f(a)* merupakan kumpulan *fitness* setiap individu pada satu populasi. Hal ini menunjukkan terjadinya konvergensi individu seperti pada Gambar 3.8 jika jumlah individu = 3 dan konvergensi dapat dijadikan salah satu kondisi terminasi.

$$\max f(a) - \min f(a) = 0 \qquad \textbf{(3.2)}$$

Individu 1	4	8	3	1	7	2
Individu 2	4	8	3	1	7	2
Individu 3	4	8	3	1	7	2

Gambar 3.8 Ilustrasi Konvergensi Individu

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi kode program dilakukan menggunakan bahasa Python.

4.1 Lingkungan Implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi ini ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Perancangan Perangkat Lunak

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i7-5500U CPU @ 2.40GHz 2.40GHz Memori: 8.00 GB
Perangkat lunak	Sistem Operasi: Microsoft Windows 10 Enterprise 64-bit Perangkat Pengembang: PyCharm, ArcGIS 10.3

4.2 Implementasi

Sub bab implementasi ini menjelaskan tentang implementasi proses yang sudah dijelaskan pada bab desain perangkat lunak.

4.2.1 Analisis Spasial

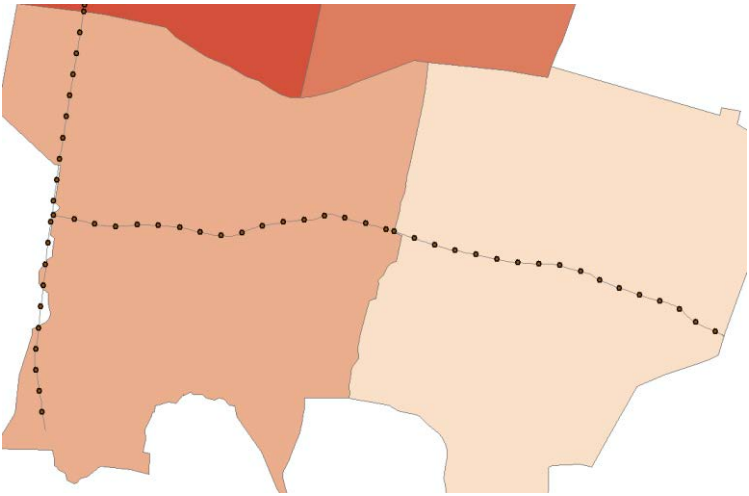
Tahap ini merupakan implementasi kegiatan analisis spasial ke dalam *tools* yang akan dibuat yang meliputi proses-proses berikut: konversi *polyline to point*, *buffering*, *erasing*, *intersect*, *distance analysis*, dan Pembaruan atribut.

4.2.1.1 Konversi Polyline to Point

Sesuai dengan penjelasan pada bab perancangan, tahap ini merupakan tahap pembangkitan domain solusi. Domain solusi didapatkan dari masukan jalan yang akan dikonversi ke dalam bentuk geometri *point*. Konveri dilakukan dengan fungsi *arcpy.CreatePointsLines_CreatePointLines()* seperti ditunjukkan pada Kode Sumber 4.1. Fungsi akan membentuk *point* di sepanjang jalan dengan interval 100 meter. Hasil konversi sepanjang jalan ditunjukkan dalam Gambar 4.1.

```
1.  arcpy.ImportToolbox("D:/BahanAjar/TugasAkhir/
    SejuTA/Code(MATLAB)/CreatePointsLines.tbx")
    messages.addMessage("CREATING POINTS ALONG
2.  LINES FROM " + jalan + " ...")
    arcpy.CreatePointsLines_CreatePointsLines(jal
3.  an, "BEGINNING", "INTERVAL", "NO", "", "100",
    "START", jalanPoint)
```

Kode Sumber 4.1 Implementasi Konversi *Polyline to Point*



Gambar 4.1 Hasil Konversi *Polyline to Point*

4.2.1.2 Buffering

Setelah mendapatkan domain solusi awal berbentuk *point* hasil dari konversi jalan maka berikutnya adalah melakukan buffer pada *feature point* lokasi rumah sakit dan sekolah di Surabaya Barat. Fungsi *arcpy.Buffer_analysis()* pada Kode Sumber 4.2 melakukan *buffer* pada *feature point* rumah sakit dan sekolah di Surabaya Barat dengan jarak 150m, sehingga menghasilkan *polygon buffer* yang luasnya 150m ke seluruh arah di sekitar rumah sakit dan sekolah seperti ditunjukkan pada Gambar 4.2.

```
1.  messages.addMessage("BUFFERING " + sekolah
    +"..." )
2.  arcpy.Buffer_analysis(sekolah, sekolahBuffer,
    "150 Meters", "FULL", "ROUND", "NONE", "",
    "PLANAR")
3.  messages.addMessage("BUFFERING " + rs +"..."
    )
4.  arcpy.Buffer_analysis(rs, rsBuffer, "150
    Meters", "FULL", "ROUND", "NONE", "",
    "PLANAR")
```

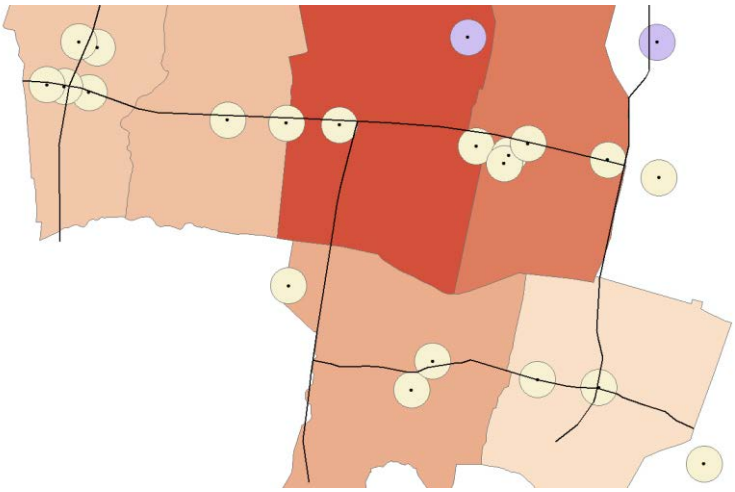
Kode Sumber 4.2 Implementasi *Buffer*

4.2.1.3 Erasing

Tahap berikutnya setelah mendapatkan polygon buffer dari rumah sakit dan sekolah maka dilakukan *overlay* untuk menghapus (*erase*). *Erase* ini dilakukan terhadap domain solusi dengan *polygon buffer* rumah sakit dan *polygon buffer* sekolah. Hal ini dilakukan untuk menghapus beberapa domain solusi yang termasuk dalam radius 150 meter dengan rumah sakit dan sekolah.

Erase diimplementasikan dengan memanggil fungsi *arcpy.Erase_analysis()*. Parameter pertama pada fungsi *arcpy.Erase_analysis()* adalah *feature class* yang akan di-*erase* dengan parameter kedua. Pada Kode Sumber 4.3 ditunjukkan bahwa operator *erase* akan dilakukan berturut-turt terhadap *jalanPoint* dan *jalanPoint_sekolahBuffer_Erase* dengan

sekolahBuffer dan *rsBuffer* yang merupakan *buffer* sejauh 150 meter dari rumah sakit dan sekolah.



Gambar 4.2 Hasil *Buffering Point* Rumah Sakit dan Sekolah

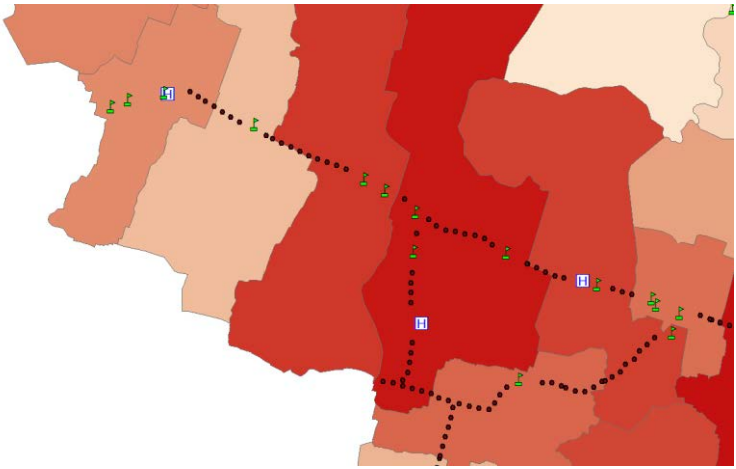
```
1.  messages.addMessage("ERASING " + jalanPoint +
    "BY " + sekolahBuffer)
2.  arcpy.Erase_analysis(jalanPoint,
    sekolahBuffer,
    jalanPoint_sekolahBuffer_Erase, "")
3.  messages.addMessage("ERASING " +
    jalanPoint_sekolahBuffer_Erase + "BY " +
    rsBuffer)
4.  arcpy.Erase_analysis(jalanPoint_sekolahBuffer
    _Erase, rsBuffer,
    jalanPoint_sekolahBuffer_rsBuffer_Erase, "")
```

Kode Sumber 4.3 Implementasi *Erase*

Setelah fungsi dijalankan, maka terlihat seperti pada Gambar 4.3 bahwa domain solusi telah berkurang pada daerah sekitar rumah sakit dan sekolah yang masing-masing dilambangkan dengan huruf “H” dan *icon* bendera.

4.2.1.4 Distance Analysis

Analisis perhitungan jarak dilakukan dengan memanggil fungsi *arcpy.PointDistance_analysis()*. Seperti ditunjukkan pada Kode Sumber 4.4, perhitungan analisis jarak dilakukan terhadap parameter 1 dan parameter 2 yaitu *jalanPoint_sekolahBuffer_rsBuffer_Erase* yang merupakan domain solusi dengan *spbu* yang merupakan sebaran SPBU eksisting.



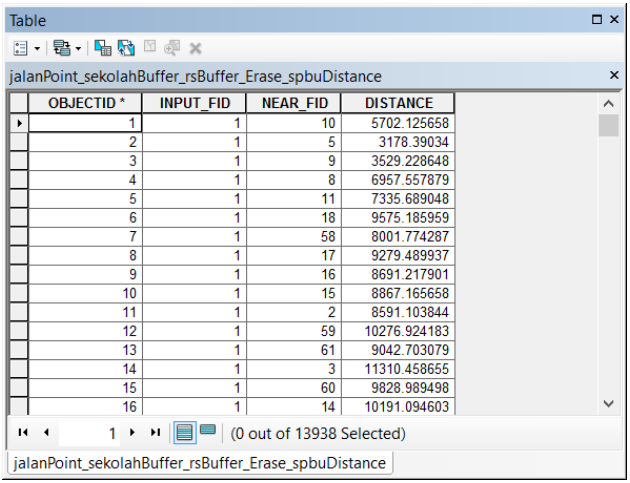
Gambar 4.3 Hasil *Erase Buffer* dengan Domain Solusi

```
1.  messages.addMessage("CALCULATING
    DISTANCE...")
2.  arcpy.PointDistance_analysis(jalanPoint_sekol
    ahBuffer_rsBuffer_Erase, spbu,
    jalanPoint_sekolahBuffer_rsBuffer_Erase_spbuD
    istance, "")
3.  arcpy.PointDistance_analysis(jalanPoint_sekol
    ahBuffer_rsBuffer_Erase, rs,
    jalanPoint_sekolahBuffer_rsBuffer_Erase_rsDis
    tance, "")
4.  arcpy.PointDistance_analysis(jalanPoint_sekol
    ahBuffer_rsBuffer_Erase, sekolah,
```

```
    jalanPoint_sekolahBuffer_rsBuffer_Erase_sekol
    ahDistance, "")
```

Kode Sumber 4.4 Implementasi Menghitung Jarak

Setelah fungsi dijalankan maka akan terbentuk tabel seperti pada Gambar 4.4. *OBJECTID* merupakan atribut unik untuk setiap *item* pada tabel. Kemudian *INPUT_FID* dan *NEAR_FID* merupakan *id* parameter 1 yang dihitung jaraknya dengan *id* pada parameter 2 dan ditunjukkan jaraknya pada atribut *DISTANCE*.



OBJECTID *	INPUT_FID	NEAR_FID	DISTANCE
1	1	10	5702.125658
2	1	5	3178.39034
3	1	9	3529.228648
4	1	8	6957.557879
5	1	11	7335.689048
6	1	18	9575.185959
7	1	58	8001.774287
8	1	17	9279.489937
9	1	16	8691.217901
10	1	15	8867.165658
11	1	2	8591.103844
12	1	59	10276.924183
13	1	61	9042.703079
14	1	3	11310.458655
15	1	60	9828.989498
16	1	14	10191.094603

Gambar 4.4 Hasil Perhitungan Jarak

4.2.1.5 Pembaruan Atribut

Tahap ini merupakan tahap untuk memperbaharui atribut pada domain solusi agar memiliki atribut jarak dan populasi.

4.2.1.5.1 Atribut Jarak

Pembaruan atribut jarak dilakukan dengan memanggil fungsi *updateDistanceField()*. Ditunjukkan pada Kode Sumber 4.5, pemanggilan fungsi dilakukan dengan 3 parameter yaitu *pointFC*, *distanceWith*, dan *distanceTable* yang secara berturut-

turut merupakan domain solusi, *layer* yang akan didapatkan jaraknya dan tabel jarak hasil perhitungan analisis jarak pada tahap sebelumnya. Fungsi ini akan melakukan *update* pada domain solusi dengan memperbaharui nilai jarak terpendek setiap domain solusi dengan SPBU eksisting.

```
1.  def updateDistanceField(pointFC,
    distanceWith, distanceTable):
2.      arcpy.AddMessage("pointFC = " + pointFC)
3.      arcpy.AddMessage("distanceWith = " +
    distanceWith)
4.      arcpy.AddMessage("distanceTable = " +
    distanceTable)
5.      str_distanceWith =
    str(distanceWith).replace(" ","_")
6.      arcpy.env.overwriteOutput = True
7.      arcpy.AddField_management(pointFC,
    "DISTANCE_" + str_distanceWith, "FLOAT")
8.      pointFC_length =
    int(arcpy.GetCount_management(pointFC).getOut
    put(0))
9.      distanceWith_length =
    int(arcpy.GetCount_management(distanceWith).g
    etOutput(0))
10.     distanceTable_length =
    int(arcpy.GetCount_management(distanceTable).
    getOutput(0))
11.     distanceField = "DISTANCE"
12.     minDistance = list()
13.     arcpy.AddMessage(pointFC_length)
14.     arcpy.AddMessage(distanceWith_length)
15.     arcpy.AddMessage(distanceTable_length)
16.     for pointID in range(1,pointFC_length+1):
17.         arcpy.AddMessage("pointID ke " +
    str(pointID))
18.         expression = "INPUT_FID = " +
    str(pointID)
19.         searchRows =
    sorted(arcpy.da.SearchCursor(distanceTable,
    distanceField, where_clause=expression))
20.         arcpy.AddMessage(searchRows[0][0])
21.         minDistance.append(searchRows[0][0])
```

```
22.     arcpy.AddMessage(minDistance)
23.     updateRows =
arcpy.da.UpdateCursor(pointFC,["OBJECTID","DI
STANCE_" + str_distanceWith])
24.     for row in updateRows:
25.         rowID = int(row[0])
26.         row[1] = minDistance[rowID-1]
27.         updateRows.updateRow(row)
28.     del updateRows
29.     del row
```

Kode Sumber 4.5 Implementasi *Update* Atribut Jarak

4.2.1.5.2 Atribut Populasi

Pembaruan atribut populasi dilakukan dengan memanggil fungsi *updatePopulationField()*. Ditunjukkan pada Kode Sumber 4.6, pemanggilan fungsi dilakukan dengan 2 parameter yaitu *pointFC* dan *populationTable* yang merupakan domain solusi dan *layer* yang memiliki informasi kepadatan penduduk per-kecamatan. Operator *intersect* dilakukan terhadap domain solusi dengan populasi dengan tujuan setiap domain solusi akan memiliki atribut baru yaitu atribut *POPULATION* berdasarkan posisi domain solusi tersebut terhadap *layer* populasi.

```
1.  def
updatePopulationField(pointFC,populationTable
):
2.      arcpy.env.overwriteOutput = True
3.      arcpy.AddField_management(pointFC,
"POPULATION", "FLOAT")
4.      pointFC_length =
int(arcpy.GetCount_management(pointFC).getOut
put(0))
5.      populationTable_length =
int(arcpy.GetCount_management(populationTable
).getOutput(0))
6.      searchCursor =
list(arcpy.da.SearchCursor(populationTable,"P
ENDUDUK"))
7.      arcpy.AddMessage(searchCursor[0])
8.      updateRows =
arcpy.da.UpdateCursor(pointFC,["LineOID","POP
```

```
    ULATION"])
9.      for row in updateRows:
10.         if row[0] is not None:
11.             arcpy.AddMessage(int(row[0]))
12.             row[1] =
    searchCursor[int(row[0])-1][0]
13.         updateRows.updateRow(row)
14.     del updateRows
15.     del row
```

Kode Sumber 4.6 Implementasi *Update* Atribut Populasi

4.2.2 Genetic Algorithms

Pada sub bab ini akan dijelaskan mengenai implementasi GA untuk melakukan pencarian solusi terbaik dari domain solusi yang tersedia.

4.2.2.1 Representasi Data

Sub bab ini membahas mengenai implementasi tahap representasi data. Representasi data dalam Tugas akhir ini dibagi menjadi dua bagian, yang pertama adalah representasi data atribut setiap *point* yang akan dibahas di sub bab ini, dan yang kedua adalah representasi data individu dalam populasi yang akan dibahas pada bab selanjutnya.

Nilai atribut setiap *point* yang akan diolah dalam GA didapat dari fungsi *arcpy.da.SearchCursor()* yang akan mengembalikan nilai dari *field* masukan sesuai dengan permintaan *where clause* yang merupakan parameter fungsi.

```
1.  distanceField =
    ["DISTANCE_SPBU_surabaya_barat",
    "DISTANCE_RS_sby_barat",
    "DISTANCE_Sekolah_sby_barat",
                    "SHAPE@X", "SHAPE@Y",
    "POPUlATION","OBJECTID"]
2.  searchCursor =
    arcpy.da.SearchCursor(inputFC,distanceField)
3.  population = [row for row in searchCursor]
```

Kode Sumber 4.7 Implementasi Representasi Data

Kode Sumber 4.7 mendefinisikan variabel *inputFC* sebagai masukan dan *distanceField* sebagai *where clause*. Pada

akhirnya variabel *population* adalah matriks dari setiap *point* dengan atribut *DISTANCE_SPBU_surabaya_barat*, *DISTANCE_RS_sby_barat*, *DISTANCE_Sekolah_sby_barat*, *SHAPE@X*, *SHAPE@Y*, *POPUlATION*, dan *OBJECTID*.

4.2.2.2 Initial Population

Representasi data yang kedua adalah representasi individu dalam satu popoulasi. Individu direpresentasikan sebagai kumpulan dari *n* buah integer, dimana setiap integer berada dalam *range 1..m* dan dimana *n* adalah jumlah *point* dan *m* adalah jumlah total *point* hasil konversi *polyline to points*.

```
1.  def generatePopulationById(pointSet,
    spbuRange, rsRange, sekolahRange, n):
2.     endom = list()
3.     i = 1
4.     while i <= n:
5.        calonEndom = random.randrange(1, 607)
6.        if int(pointSet[calonEndom - 1][0]) in
    range(spbuRange[0], spbuRange[1]) and int(
                pointSet[calonEndom - 1][1]) in
    range(rsRange[0], rsRange[1]) and int(
                pointSet[calonEndom - 1][2]) in
    range(sekolahRange[0], sekolahRange[1]):
7.           if calonEndom not in endom:
8.              endom.append(calonEndom)
                i = i + 1
9.     return endom
10.
```

Kode Sumber 4.8 Implementasi Inisialisasi Populasi (1)

Kode Sumber 4.8 melakukan pembangkitan individu random dengan memanggil fungsi *generatePopulationById()*. Parameter *pointSet* adalah masukan data *point* hasil konversi yang akan digunakan untuk membatasi masukan dengan batasan bahwa *point* yang terpilih untuk menjadi individu, masing-masing atribut *DISTANCE_SPBU_surabaya_barat*, *DISTANCE_RS_sby_barat*, *DISTANCE_Sekolah_sby_barat* berada pada *range spbuRange*, *rsRange*, *sekolahRange*.

Pemilihan *point* pada fungsi ini dilakukan sebanyak *n* kali, yang artinya individu akan menghasilkan *point* sebanyak *n*.

```
1.  def createInitialPopulation(population,
    spbuRange, rsRange, sekolahRange, m, n):
2.      for i in range(1, m + 1):
3.          individual =
    generatePopulationById(population,
    spbuRange, rsRange, sekolahRange, n)
4.          IP.append(individual)
        print IP
5.      return IP
```

Kode Sumber 4.9 Implementasi Inisialisasi Populasi (2)

Kemudian populasi awal dihasilkan dengan memanggil fungsi *createInitialPopulation()*. Dalam Kode Sumber 4.9, fungsi ini memanggil fungsi sebelumnya yaitu *generatePopulationById()* sebanyak *m* kali dan mengembalikan datanya ke variabel *IP* sehingga variabel *IP* berisi populasi awal yang terdiri dari *m* individu.

4.2.2.3 Fitness Evaluation

Pada tahap ini, populasi awal yang sudah terbentuk pada tahap sebelumnya dihitung nilai *fitness* setiap individunya.

```
1.  def calculateFitness(IP):
2.      fitnessList = list()
3.      for i in range(1, m + 1):
4.          spbuDistSum = 0
5.          pendudukSum = 0
6.          coordinateList = list()
7.          for j in range(1, n + 1):
8.              spbuDistSum += population[IP[i -
    1][j - 1] - 1][0]
9.              coordinate = np.array(
                    [population[IP[i - 1][j - 1]
    - 1][3], population[IP[i - 1][j - 1] -
10. 1][4]])
11.             coordinateList.append(coordinate)
                pendudukSum += population[IP[i -
12. 1][j - 1] - 1][5]
```

```
13.         individualFitList = list()
14.         eucDistList = [np.linalg.norm(a - b)
    for a, b in combinations(coordinateList, 2)]
15.         individualFitList.append(round
    (spbuDistSum, 2))
16.         individualFitList.append(round
    (sum(eucDistList), 2))
17.         individualFitList.append(pendudukSum)
18.         fitnessList.append(individualFitList)
19.     fitnessList = minmaxNorm(fitnessList)
20.     return weightedSum(fitnessList)
```

Kode Sumber 4.10 Implementasi Perhitungan *Fitness* (1)

Fungsi *calculateFitness()* pada Kode Sumber 4.10 menghasilkan *array* yang berisi jumlah total jarak setiap *point* dengan SPBU terdekat, jumlah nilai *Euclidean Distance* setiap *point*, dan jumlah populasi dalam satu individu yang diiterasi sebanyak jumlah individu dalam satu populasi dan kemudian *array* tersebut dimasukkan ke dalam variabel *fitnesslist*. Selanjutnya *fitnesslist* dinormalisasi dengan memanggil fungsi *minmaxNorm()* pada Kode Sumber 4.11 dengan variabel *fitnessList* sebagai parameternya.

```
1.  def minmaxNorm(matrix):
2.      criteriaSPBU = [matrix[i - 1][0] for i in
    range(1, len(matrix) + 1)]
3.      criteriaEucDist = [matrix[i - 1][1] for i
    in range(1, len(matrix) + 1)]
4.      criteriaPenduduk = [matrix[i - 1][2] for
    i in range(1, len(matrix) + 1)]
5.      ubSPBU = max(criteriaSPBU)
6.      lbSPBU = min(criteriaSPBU)
7.      ubEucDist = max(criteriaEucDist)
8.      lbEucDist = min(criteriaEucDist)
9.      ubPenduduk = max(criteriaPenduduk)
10.     lbPenduduk = min(criteriaPenduduk)
11.     for i in range(1, len(matrix) + 1):
12.         if int(ubSPBU - lbSPBU) == 0 or
    int(ubPenduduk - lbPenduduk) == 0 or
    int(ubEucDist - lbEucDist) == 0:
13.             print "reach best generation"
14.             print quit()
```

```
15.         matrix[i - 1][0] = ((matrix[i - 1][0]
    - lbSPBU) / (ubSPBU - lbSPBU) * (1 - 0)) + 0
16.         matrix[i - 1][1] = ((matrix[i - 1][1]
    - lbEucDist) / (ubEucDist - lbEucDist) * (1 -
    0)) + 0
17.         matrix[i - 1][2] = ((matrix[i - 1][2]
    - lbPenduduk) / (ubPenduduk - lbPenduduk) *
    (1 - 0)) + 0
18.     return matrix
```

Kode Sumber 4.11 Implementasi Normalisasi Data

Pada fungsi *minmaxNorm()*, normalisasi *minmax* dilakukan terhadap data *fitnessList* dengan tujuan mendapatkan data dengan *range* penyebaran data yang sama yaitu 0..1 untuk menghindari dominasi nilai oleh salah satu atribut. Fungsi ini akan mengembalikan *fitnessList* yang sudah ternormalisasi. Selanjutnya, *fitnessList* yang sudah ternormalisasi akan dihitung nilai *weighted sum* nya melalui fungsi *weightedSum()* seperti ditunjukkan pada Kode Sumber 4.12.

```
1.  def weightedSum(matrix):
2.      hasil = list()
        for i in range(1, len(matrix) + 1):
3.          matrix[i - 1][0] = matrix[i - 1][0] *
    weight[0]
4.          matrix[i - 1][1] = matrix[i - 1][1] *
    weight[1]
5.          matrix[i - 1][2] = matrix[i - 1][2] *
6.  weight[2]
7.          hasil.append([i, sum(matrix[i - 1])])
8.      return hasil
```

Kode Sumber 4.12 Implementasi Perhitungan *Fitness* (2)

4.2.2.4 Reproduce

Tahap ini bertujuan untuk menghasilkan individu baru hasil penyilangan dari 2 *parent*. Dalam Kode Sumber 4.13 untuk mendapatkan *parent*, pertama-tama individu di dalam *fitnessList* perlu diurutkan dari yang tertinggi hingga terendah. Kemudian proses seleksi dilakukan dengan memanggil fungsi *rankSelection()* yang akan mengembalikan nilai *index* dari

individu yang terpilih sebagai *parent*. Setelah *parent* didapatkan, fungsi *uniformCrossover()* dipanggil. Fungsi ini akan mengembalikan hasil penyilangan dua *parent* dengan metode *uniform crossover*.

```
1.  def reproduce(IP):
2.      fitnessList =
    sorted(calculateFitness(IP), key=lambda x:
    x[1], reverse=True)
3.      parent1 = fitnessList[rankSelection(m)]
4.      parent2 = fitnessList[rankSelection(m)]
5.      return uniformCrossover(n, parent1,
    parent2)
```

Kode Sumber 4.13 Implementasi Tahap Reproduksi

4.2.2.5 Selection

Fungsi *rankSelection()* dipanggil untuk menentukan individu mana yang akan menjadi *parents*. Ditunjukkan pada Kode Sumber 4.14 penentuan *parents* dilakukan dengan cara me-random nilai antara 0..1. Jika nilai hasil random kurang dari nilai variabel *choosebest*, maka fungsi akan mengembalikan nilai random antara 0 hingga *m* x *chooseparentrate*, yang artinya akan memilih salah satu dari *m* x *chooseparentrate* individu terbaik.

```
1.  def rankSelection(m, choosebest=0.6,
2.  chooseparentrate=0.6):
3.      if random.random() > choosebest:
4.          return random.randint(0, m *
    chooseparentrate)
5.      else:
6.          return random.randint(m *
    chooseparentrate, m - 1)
```

Kode Sumber 4.14 Implementasi Tahap Seleksi

Jika nilai hasil random lebih dari nilai variabel *choosebest*, maka fungsi akan mengembalikan nilai random antara *m* x *chooseparentrate* hingga *m*-1, yang artinya akan memilih salah satu dari *m* x (1-*chooseparentrate*) individu terburuk, dimana *m* adalah jumlah individu dalam satu populasi.

4.2.2.6 Crossover

Fungsi *uniformCrossover()* pada Kode Sumber 4.15 menyilangkan 2 *parent*, *parent1* dan *parent2* dengan metode *uniform crossover*. *Uniform Crossover* menggunakan array *mask* sebanyak *n* yang berisi nilai 0 dan 1.

```
1.  def uniformCrossover(n, parent1, parent2):
2.      child1 = list()
3.      child2 = list()
4.      mask = mutate([random.randint(0, 1) for
    i in range(1, n + 1)])
5.      for i in range(1, n + 1):
6.          if mask[i - 1] == 0:
7.              c1 = IP[parent1[0] - 1][i - 1]
8.              c2 = IP[parent2[0] - 1][i - 1]
9.          else:
10.             c1 = IP[parent2[0] - 1][i - 1]
11.             c2 = IP[parent1[0] - 1][i - 1]
12.         child1.append(c1)
13.         child2.append(c2)
14.     return child1, child2
```

Kode Sumber 4.15 Implementasi Tahap Tukar Silang

Jika pada *index* ke-*i* nilai *mask*[*i*] = 0, maka *child1*[*i*] = *parent1*[*i*] dan *child2*[*i*] = *parent2*[*i*]. Jika pada *index* ke-*i* nilai *mask*[*i*] = 1, maka *child1*[*i*] = *parent2*[*i*] dan *child2*[*i*] = *parent1*[*i*].

4.2.2.7 Mutation

Fungsi *mutate()* dipanggil untuk memutasi/memodifikasi nilai di dalam *array mask* jika semua nilai di dalam *array* tersebut bernilai sama.

```
1.  def mutate(mask):
2.      if 0 not in mask or 1 not in mask:
3.          mutationChromosome =
    [random.randint(0, 1) for i in range(1, n +
    1)]
4.          if 0 not in mutationChromosome or 1
    not in mutationChromosome:
```

```
5.              mutationChromosome =
    mutate(mutationChromosome)
6.              return mutationChromosome
7.          else:
8.              return mutationChromosome
9.      else:
10.         return mask
```

Kode Sumber 4.16 Implementasi Tahap Mutasi

Dalam Kode Sumber 4.16, terlihat bahwa fungsi ini merupakan fungsi rekursif yang akan merandom nilai *mask* hingga tidak ada *mask* yang memiliki nilai yang sama pada semua elemennya.

4.2.2.8 Regenerate

Fungsi *regenerate()* pada Kode Sumber 4.17 dipanggil untuk menghasilkan generasi baru dari parameter *IP*. Parameter *newindividualrate* memberikan kesempatan kepada sebanyak *newindividualrate* x *m* individu terbaik pada populasi untuk menjadi individu pada generasi selanjutnya.

Sisa individu sebanyak (1 – *newindividualrate*) x *m* pada generasi baru diisi oleh hasil dari penyilangan dua *parent*. Fungsi ini mengembalikan *array newgeneration* yang berisi generasi baru.

4.2.2.9 Termination

Parameter *termination* terdapat pada kode sumber *minmaxNorm()*. Program berhenti dan dinyatakan telah mencapai generasi terbaik jika memenuhi hal-hal berikut:

- *oubSPBU = olbSPBU*
- *oubPenduduk = olbPenduduk*
- *oubEucDist = olbEucDist*

Dimana *oubSPBU*, *oubPenduduk*, dan *oubEucDist* masing-masing adalah batas nilai tertinggi untuk setiap atribut, dan *olbSPBU*, *olbPenduduk*, dan *olbEucDist* masing-masing adalah batas nilai terendah untuk setiap atribut.

```
1.  def regenerate(IP, newindividualrate=0.6):
2.      newgeneration = list()
3.      populationFitness =
    sorted(calculateFitness(IP), key=lambda x:
    x[1], reverse=True)
4.      for i in range(1, int(newindividualrate *
    len(IP)) + 1):

5.  newgeneration.append(IP[populationFitness[i -
    1][0] - 1])
6.      for i in range(1, int(((1 -
    newindividualrate) * len(IP) / 2) + 1)):
7.          child1, child2 = reproduce(IP)
8.          newgeneration.append(child1)
9.          newgeneration.append(child2)
10.     return newgeneration
```

Kode Sumber 4.17 Implementasi Tahap Regenerasi

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan hasil uji coba dan evaluasi program yang telah selesai diimplementasi.

5.1 Lingkungan Pelaksanaan Uji Coba

Lingkungan uji coba yang akan digunakan adalah:

1. Perangkat Keras
   Prosesor Intel® Core™ i7-5500U CPU @ 2.40GHz 2.40GHz RAM 8 GB.
   Sistem Operasi 64-bit .
2. Perangkat Lunak
   Sistem Operasi Microsoft Windows 8 64-bit Pro.
   Perangkat Lunak ArcGIS 10.3.
   Perangkat Pengembang PyCharm.

5.2 Skenario Pengujian Pencarian Lokasi Optimal

Pada sub bab ini akan dijelaskan mengenai skenario uji coba yang telah dilakukan. Penulis telah melakukan beberapa skenario uji coba, diantaranya yaitu:

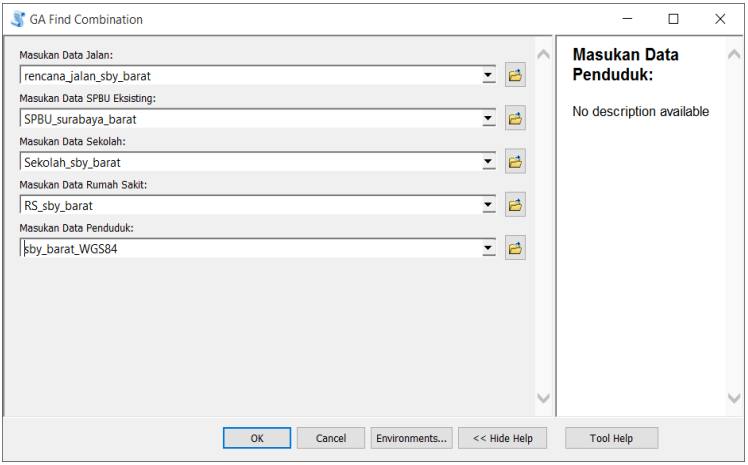
1. Pengujian pencarian alternatif lokasi SPBU. Pengujian ini menggunakan data-data SPBU eksisting, rumah sakit, sekolah, sebaran populasi, dan rencana jalan yang telah ada sebelumnya sebagai parameter.
2. Perbandingan *Fitness Growth* berdasarkan variasi parameter. Parameter yang akan diuji antara lain adalah jumlah individu dalam 1 populasi dan besaran interval pada proses konversi *polyline to point* yang akan berpengaruh pada jumlah domain solusi.
3. Pengujian nilai konsistensi hasil pencarian alternatif lokasi SPBU. Pengujian ini dilakukan untuk mengetahui performa *toolbox* dengan menghitung nilai konsistensi hasil pencarian alternatif lokasi SPBU.

4. Pengujian evaluasi hasil. Pengujian ebaluasi hasil dilakukan dengan membandingkan hasil pencarian alternatif lokasi menggunakan GA dengan hasil *clustering* menggunakan *Grouping Analysis tools* yang telah tersedia pada ArcGIS 10.3.

Setiap pengujian pada skenario pengujian dilakukan dengan mencari 5 lokasi alternatif terbaik.

5.2.1 Pengujian Pencarian Alternatif Lokasi SPBU

Pengujian pada tahap ini merupakan pengujian pencarian alternatif lokasi SPBU optimal. Pengujian dilakukan dengan pencarian 5 titik alternatif lokasi SPBU optimal. Uji coba dilakukan dengan menjalankan *Python Toolbox* yang sudah dibuat pada lingkungan ArcGIS 10.3. Gambar 5.1 menunjukkan antarmuka *Python Toolbox* ketika dijalankan.



Gambar 5.1 Antarmuka *Python Toolbox*

Pada Gambar 5.1 ditunjukkan bahwa yang menjadi masukan pada *toolbox* secara berturut-turut adalah data-data sebagai berikut:

1. *rencana_jalan_sby_barat*

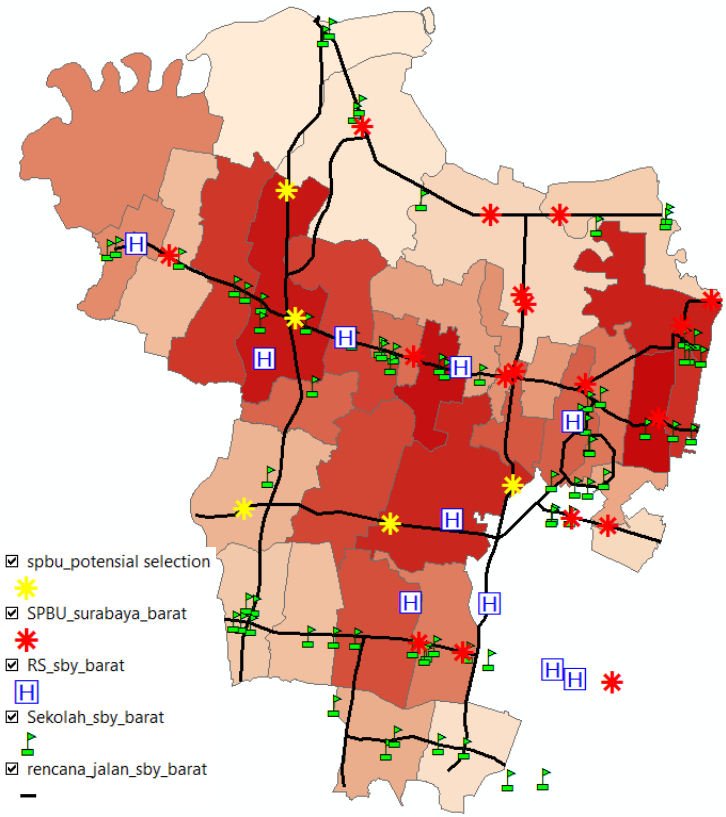
2. *SPBU_surabaya_barat*
3. *Sekolah_sby_barat*
4. *RS_sby_barat*
5. *sby_barat_WGS84*

Pada pengujian ini, terdapat beberapa skenario pengujian yaitu dengan modifikasi data jumlah penduduk pada *layer sby_barat_WGS84* yang memiliki informasi mengenai jumlah penduduk per kelurahan. Skenario pengujian diperlihatkan pada Tabel 5.1. Pengujian perubahan data jumlah penduduk ini dilakukan untuk menguji apakah sistem dapat mencari lokasi yang relatif bagus dengan perubahan jumlah penduduk pada setiap percobaannya.

Tabel 5.1 Skenario Pengujian Pencarian Lokasi

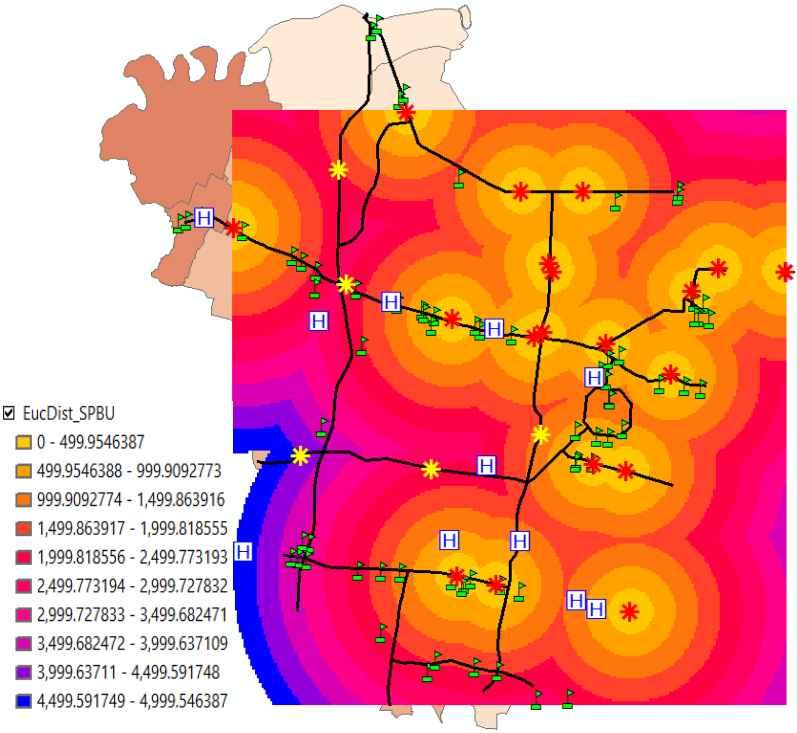
Skenario	Perubahan	Jumlah Penduduk
1	-	-
2	Kel. Made	2000
3	Kel. Lakarsantri	15000
4	Kel. Sememi	2000

Pada pengujian skenario 1, tidak terdapat perubahan pada data jumlah penduduk. Setelah *toolbox* dijalankan, maka hasil pencarian alternatif lokasi SPBU pada skenario 1 akan memberikan alternatif lokasi SPBU berupa *layer* pada *workspace* dengan tipe geometri *point* sebagaimana ditunjukkan pada Gambar 5.2.



Gambar 5.2 Hasil Pencarian Lokasi SPBU Skenario 1

Hasil pencarian alternatif lokasi SPBU ditunjukkan dengan logo bintang berwarna kuning. Jika dibandingkan dengan Gambar 5.3, dapat dilihat bahwa dalam pencarian 5 lokasi alternatif, 4 alternatif lokasi mengarah kepada wilayah dengan kepadatan penduduk yang relatif tinggi namun jarak dengan SPBU eksisting bukan jarak yang maksimal dan 1 alternatif mengarah pada daerah dengan jarak yang maksimal namun berada di wilayah dengan populasi yang bukan maksimal.

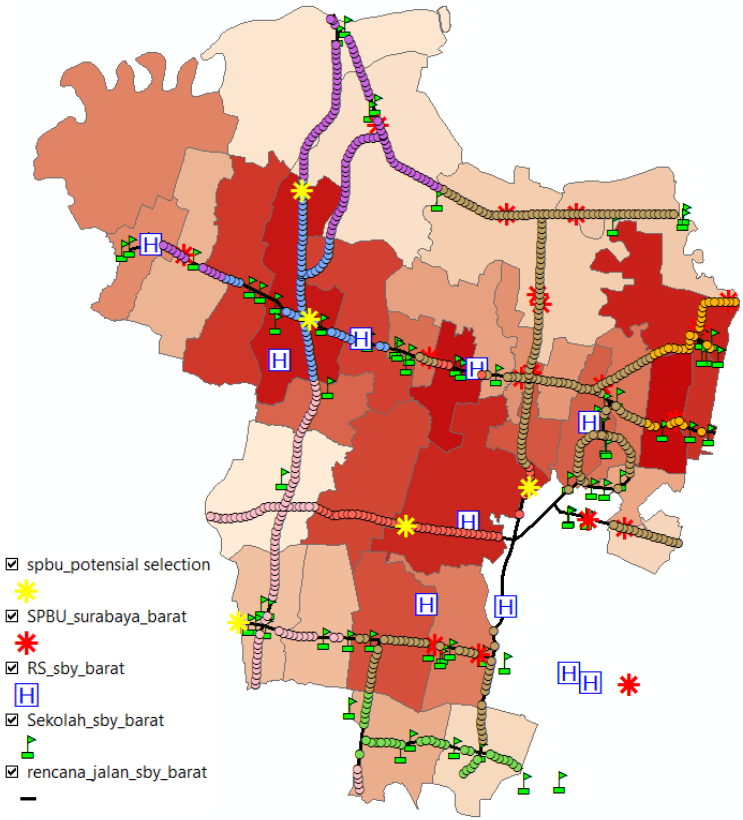


Gambar 5.3 Analisis *Euclidean Distance* Skenario 1

Gambar 5.3 menunjukkan hasil analisis *Euclidean Distance* pada *layer* SPBU eksisting. Warna semakin biru menunjukkan bahwa jarak dengan SPBU semakin jauh dan sebaliknya warna semakin kuning menunjukkan jarak dengan SPBU semakin dekat. Terlihat bahwa hasil pencarian lokasi SPBU mendapatkan lokasi-lokasi yang jaraknya relatif jauh terhadap lokasi SPBU eksisting.

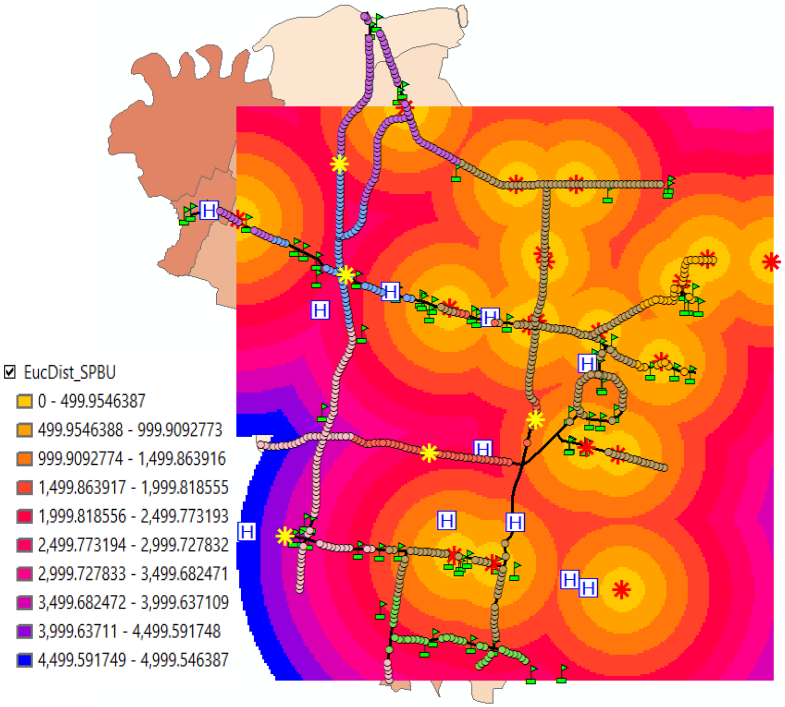
Pada pengujian skenario 2, perubahan dilakukan pada data jumlah penduduk pada kelurahan Made menjadi sebesar 2000. Hal ini untuk menunjukkan bahwa setelah terjadi perubahan data jumlah penduduk menjadi minimal pada salah satu kelurahan yang terdapat titik hasil pencarian lokasi

alternatif SPBU, maka ketika program dijalankan kembali tidak akan memberikan hasil pencarian pada kelurahan tersebut. Setelah *toolbox* dijalankan, maka hasil pencarian alternatif lokasi SPBU pada skenario 2 akan memberikan alternatif lokasi SPBU berupa *layer* pada *workspace* dengan tipe geometri *point* sebagaimana ditunjukkan pada Gambar 5.4.



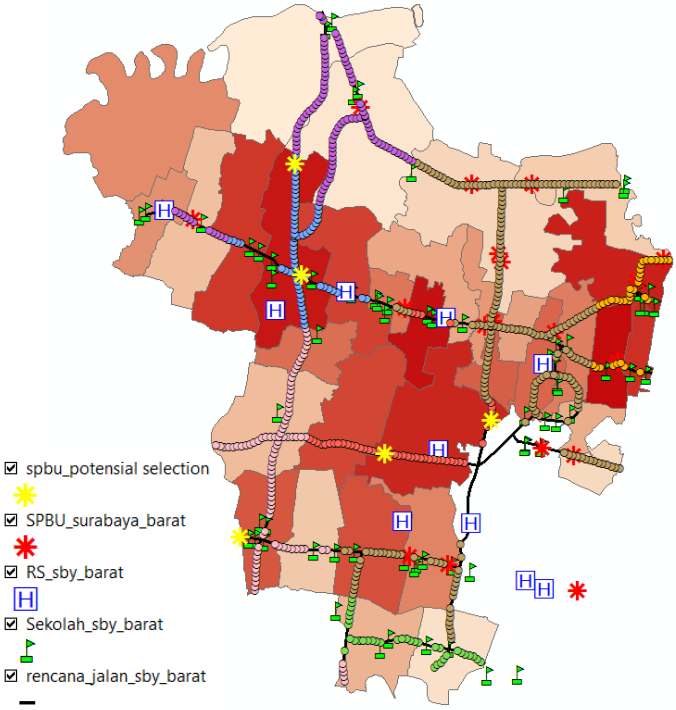
Gambar 5.4 Hasil Pencarian Lokasi SPBU Skenario 2

Berdasarkan hasil pencarian lokasi SPBU dengan skenario 2, dapat dilihat bahwa pencarian tidak memberikan hasil di kelurahan Made dengan nilai jumlah penduduk yang telah diminimalkan. Jika dibandingkan dengan hasil pencarian pada skenario 1, dapat dilihat bahwa 4 dari 5 titik hasil pencarian pada skenario 1 muncul pada hasil pencarian skenario 2 dengan 1 titik sisanya menghindari dari daerah yang telah diminimalkan jumlah penduduknya dan mengarah pada wilayah yang jarak dengan SPBU terdekat adalah maksimum seperti ditunjukkan pada Gambar 5.5.



Gambar 5.5 Analisis *Euclidean Distance* Skenario 2

Pada pengujian skenario 3, perubahan dilakukan pada data jumlah penduduk pada kelurahan Lakarsantri menjadi sebesar 15000. Hal ini untuk menunjukkan bahwa setelah terjadi perubahan data jumlah penduduk menjadi sebesar 15000 dari 7500 yang mana lebih besar jika dibandingkan dengan kelurahan Made pada skenario 1, yang menjadi salah satu kelurahan yang terdapat titik hasil pencarian lokasi alternatif SPBU, maka ketika program dijalankan kembali seharusnya tidak akan memberikan hasil pencarian pada kelurahan tersebut dan seharusnya memberikan titik hasil pencarian pada kelurahan Lakarsantri.



Gambar 5.6 Hasil Pencarian Lokasi SPBU Skenario 3

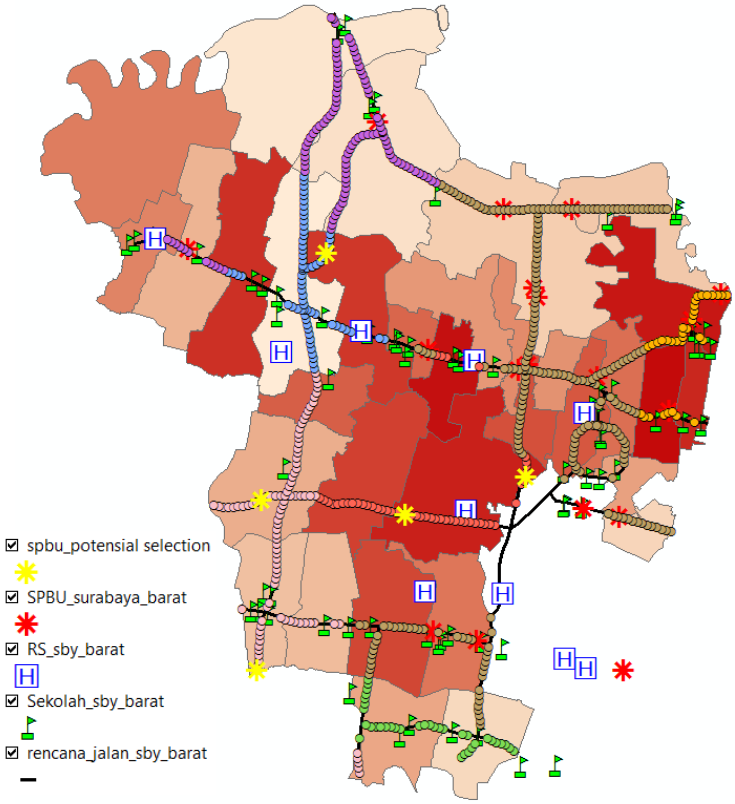
Setelah *toolbox* dijalankan, maka hasil pencarian alternatif lokasi SPBU pada skenario 3 memberikan alternatif lokasi SPBU berupa *layer* pada *workspace* dengan tipe geometri *point* sebagaimana ditunjukkan pada Gambar 5.6. Berdasarkan hasil pencarian skenario 3 dapat dilihat bahwa pencarian lokasi SPBU berhasil menemukan lokasi sesuai yang diharapkan yaitu 4 titik sesuai dengan hasil pada skenario 1 dan 1 titik menuju kelurahan Lakarsantri yang jumlah penduduknya ditingkatkan menjadi 15000.

Kemudian selanjutnya pada pengujian skenario 4, perubahan dilakukan pada data jumlah penduduk pada kelurahan Sememi menjadi sebesar 2000. Hal ini untuk menunjukkan bahwa setelah terjadi perubahan data jumlah penduduk menjadi hanya 2000 dari data jumlah penduduk sebelumnya yaitu sebesar 30000 pada skenario 1, maka ketika program dijalankan kembali seharusnya tidak akan memberikan hasil pencarian pada kelurahan tersebut karena memiliki jumlah penduduk yang minimal.

Setelah *toolbox* dijalankan, maka hasil pencarian alternatif lokasi SPBU pada skenario 4 memberikan alternatif lokasi SPBU berupa *layer* pada *workspace* dengan tipe geometri *point* sebagaimana ditunjukkan pada Gambar 5.7. Berdasarkan hasil pencarian skenario 4 dapat dilihat bahwa pencarian lokasi SPBU berhasil menemukan lokasi sesuai yang diharapkan yaitu pencarian tidak memberikan hasil pencarian ke daerah keluraha Sememi yang jumlah penduduknya telah menjadi minimal.

Berdasarkan hasil pengujian pencarian lokasi SPBU dengan berbagai skenario dapat disimpulkan bahwa sistem dapat mencari alternatif lokasi SPBU dengan baik. Hal ini ditunjukkan dengan perubahan hasil pencarian lokasi ketika data masukan berupa data jumlah penduduk berubah. Jika sebuah daerah diberikan nilai jumlah penduduk yang minimal maka sistem tidak memberikan hasil pencarian lokasi SPBU di daerah tersebut, dan ketika sebuah daerah yang memiliki jarak

relatif jauh dari SPBU terdekat dan nilai jumlah penduduknya ditinggikan, maka sistem akan memberikan hasil pencarian di daerah tersebut.



Gambar 5.7 Hasil Pencarian Lokasi SPBU Skenario 4

5.2.2 Pengujian *Fitness Growth*

Pengujian pada tahap ini merupakan pengujian terhadap nilai *fitness* setiap generasi yang dihasilkan. Pengujian ini dilakukan dengan percobaan nilai 2 parameter yaitu nilai jumlah individu dalam satu populasi dan besaran interval pada proses konversi *polyline to point* yang akan berpengaruh pada jumlah domain solusi. Besaran interval berkolerasi dengan nilai *xy tolerance* sebagaimana ditunjukkan pada Persamaan 5.1. Nilai *xy tolerance* adalah nilai toleransi daerah dari nilai koordinat domain solusi.

$$xy\ tolerance = \frac{interval}{2} \quad \textbf{(5.1)}$$

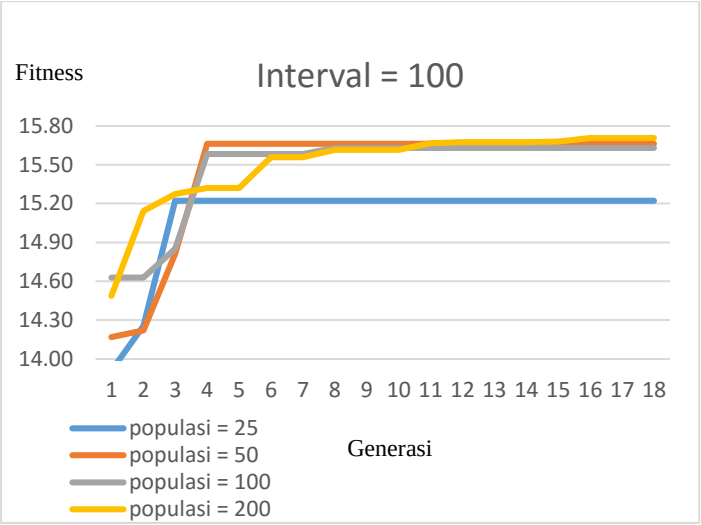
Jumlah individu yang akan diuji coba adalah 25, 50, 100, 200 dan besaran interval yang akan diuji coba adalah 100,500,1000. Skenario pengujian nilai parameter ditunjukkan pada Tabel 5.2.

Tabel 5.2 Skenario Pengujian *Fitness Growth*

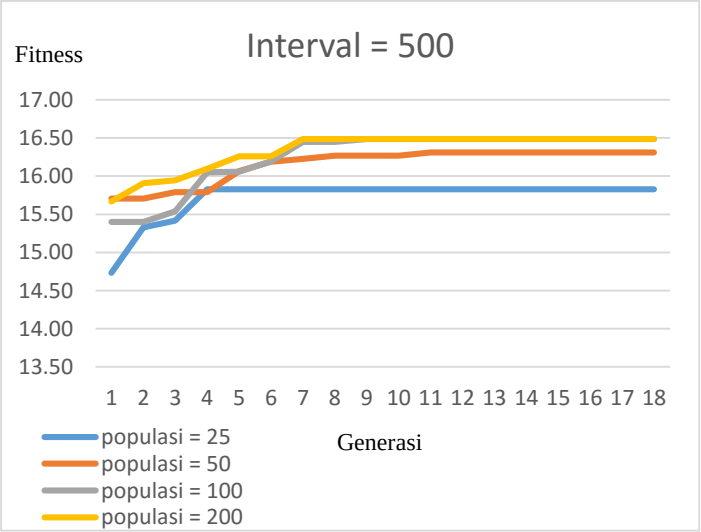
Skenario	Interval	Jumlah Individu
1	100	25
		50
		100
		200
2	500	25
		50
		100
		200
3	1000	25
		50
		100
		200

Tabel 5.3 Hasil Pengujian *Fitness Growth* Skenario 1

Generasi	Fitness			
	Jumlah Individu *= 25*	Jumlah Individu *= 50*	Jumlah Individu *= 100*	Jumlah Individu *= 200*
1	13,91	14,17	14,63	14,49
2	14,26	14,22	14,63	15,14
3	15,22	14,81	14,85	15,27
4	15,22	15,66	15,58	15,32
5	15,22	15,66	15,58	15,32
6	15,22	15,66	15,58	15,56
7	15,22	15,66	15,58	15,56
8	15,22	15,66	15,63	15,62
9	15,22	15,66	15,63	15,62
10	15,22	15,66	15,63	15,62
11	15,22	15,66	15,63	15,67
12	15,22	15,66	15,63	15,68
13	15,22	15,66	15,63	15,68
14	15,22	15,66	15,63	15,68
15	15,22	15,66	15,63	15,68
16	15,22	15,66	15,63	15,71
17	15,22	15,66	15,63	15,71
18	15,22	15,66	15,63	15,71



Gambar 5.8 Grafik Hasil Pengujian *Fitness Growth* Skenario 1



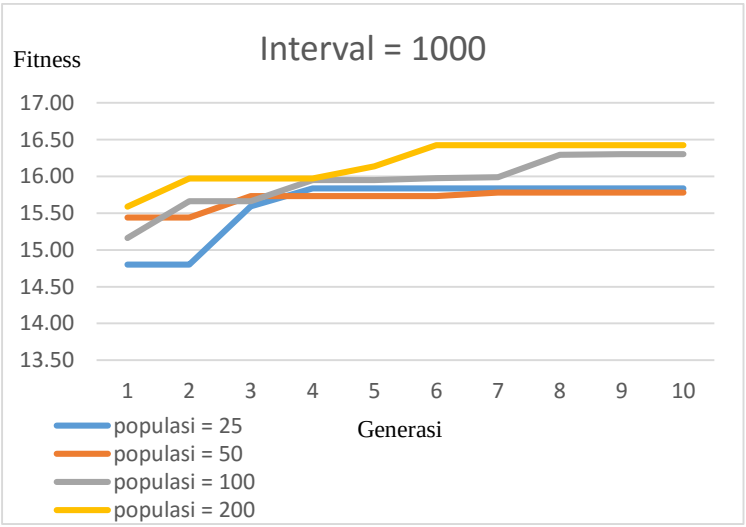
Gambar 5.9 Grafik Hasil Pengujian *Fitness Growth* Skenario 2

Tabel 5.4 Hasil Pengujian *Fitness Growth* Skenario 2

Generasi	Fitness			
	Jumlah Individu *= 25*	Jumlah Individu *= 50*	Jumlah Individu *= 100*	Jumlah Individu *= 200*
1	14,73	15,71	15,40	15,67
2	15,33	15,71	15,40	15,91
3	15,41	15,79	15,54	15,94
4	15,83	15,79	16,05	16,10
5	15,83	16,07	16,06	16,26
6	15,83	16,19	16,19	16,26
7	15,83	16,23	16,45	16,49
8	15,83	16,27	16,45	16,49
9	15,83	16,27	16,49	16,49
10	15,83	16,27	16,49	16,49
11	15,83	16,31	16,49	16,49
12	15,83	16,31	16,49	16,49
13	15,83	16,31	16,49	16,49
14	15,83	16,31	16,49	16,49
15	15,83	16,31	16,49	16,49
16	15,83	16,31	16,49	16,49
17	15,83	16,31	16,49	16,49
18	15,83	16,31	16,49	16,49

Tabel 5.5 Hasil Pengujian *Fitness Growth* Skenario 3

Generasi	Fitness			
	Jumlah Individu *= 25*	Jumlah Individu *= 50*	Jumlah Individu *= 100*	Jumlah Individu *= 200*
1	14,80	15,44	15,16	15,59
2	14,80	15,44	15,66	15,97
3	15,59	15,73	15,66	15,97
4	15,84	15,73	15,95	15,97
5	15,84	15,73	15,95	16,14
6	15,84	15.73	15,98	16,43
7	15,84	Generasi	15,99	16,43
8	15,84	15,78	16,30	16,43
9	15,84	15,78	16,30	16,43
10	15,84	15,78	16,30	16,43



Gambar 5.10 Grafik Hasil Pengujian *Fitness Growth* Skenario 3

Hasil pengujian skenario pertama ditunjukkan pada Tabel 5.3 dan *fitness growth* ditunjukkan pada Gambar 5.8. Kemudian hasil pengujian skenario kedua ditunjukkan pada Tabel 5.4 dan *fitness growth* ditunjukkan pada Gambar 5.9. Terakhir, hasil pengujian skenario ketiga ditunjukkan pada Tabel 5.5 dan *fitness growth* ditunjukkan pada Gambar 5.10.

Dari setiap hasil uji coba terlihat bahwa nilai *fitness* mencapai nilai relatif besar pada skenario 2 dengan jumlah individu 100 dan 200, serta pada skenario 3 dengan jumlah individu 200. Berdasarkan hasil pengujian ini, parameter tersebut akan dijadikan parameter sebagai uji coba pada pengujian selanjutnya.

5.2.3 Pengujian Nilai Konsistensi

Pada bagian ini, untuk mendapatkan nilai konsistensi, perhitungan dilakukan dengan menghitung rasio similaritas hasil pencarian alternatif lokasi SPBU untuk setiap parameter d berdasarkan pada pengujian parameter pada sub bab sebelumnya. Perhitungan rasio similaritas dilakukan dengan membandingkan hasil alternatif lokasi SPBU sebanyak 10 kali percobaan menggunakan perhitungan *string matching*. Pengujian dilakukan dengan menjalankan *toolbox* sebanyak 10 kali untuk setiap skenario parameter pada Tabel 5.6, kemudian hasil pencarian lokasi dibandingkan untuk setiap percobaan.

Tabel 5.6 Skenario Pengujian Nilai Konsistensi

Skenario	interval	jumlah individu
1	500	100
2	500	200
3	1000	200

Tabel 5.7 Hasil Pengujian Konsistensi Skenario 1

Percobaan	Alternatif ke-				
	1	2	3	4	5
1	1	35	37	68	107
2	24	32	39	79	96
3	33	42	46	68	96
4	9	32	42	79	96
5	33	37	46	79	96
6	35	37	46	79	96
7	10	35	46	81	107
8	33	37	46	68	93
9	10	32	39	79	107
10	1	32	41	75	107

Tabel 5.8 Perhitungan Nilai Similaritas Skenario 1

Percobaan	1	2	3	4	5	6	7	8	9	10
1	0	0	0,2	0	0,2	0,4	0,4	0,4	0,2	0,4
2	0	0	0,2	0,6	0,4	0,4	0	0	0,6	0,2
3	0	0	0	0,4	0,6	0,4	0,2	0,6	0	0
4	0	0	0	0	0,4	0,4	0	0	0,4	0,2
5	0	0	0	0	0	0,8	0,2	0,6	0,2	0
6	0	0	0	0	0	0	0.4	0,4	0,2	0
7	0	0	0	0	0	0	0	0,2	0,4	0,2
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0,4
10	0	0	0	0	0	0	0	0	0	0

Tabel 5.7 menunjukkan hasil percobaan dengan nilai interval = 500 dan jumlah individu = 100, sedangkan Tabel 5.8 menunjukkan matriks similaritas dengan membandingkan setiap hasil percobaan pada Tabel 5.7.

Tabel 5.9 Hasil Pengujian Konsistensi Skenario 2

Percobaan	Alternatif ke-				
	1	2	3	4	5
1	33	41	46	79	107
2	35	42	46	79	96
3	35	42	46	79	96
4	29	37	40	81	107
5	35	37	46	81	107
6	35	37	46	79	96
7	29	37	40	68	93
8	24	29	39	79	96
9	13	32	39	68	93
10	35	37	46	68	96

Tabel 5.10 Perhitungan Nilai Similaritas Skenario 2

Percobaan	1	2	3	4	5	6	7	8	9	10
1	0	0,4	0,4	0,2	0,4	0,4	0	0,2	0	0,2
2	0	0	1	0	0,4	0,8	0	0,4	0	0,6
3	0	0	0	0	0,4	0,8	0	0,4	0	0,6
4	0	0	0	0	0,6	0,2	0,6	0,2	0	0,2
5	0	0	0	0	0	0,6	0,2	0	0	0,6
6	0	0	0	0	0	0	0,2	0.4	0	0,8
7	0	0	0	0	0	0	0	0,2	0,4	0,4
8	0	0	0	0	0	0	0	0	0,2	0,2
9	0	0	0	0	0	0	0	0	0	0,2
10	0	0	0	0	0	0	0	0	0	0

Tabel 5.9 menunjukkan hasil percobaan dengan nilai interval = 500 dan jumlah individu = 200, sedangkan Tabel 5.10 menunjukkan matriks similaritas dengan membandingkan setiap hasil percobaan pada Tabel 5.9.

Tabel 5.11 Hasil Pengujian Konsistensi Skenario 3

Percobaan	Alternatif ke-				
	1	2	3	4	5
1	20	23	26	42	56
2	20	23	26	42	56
3	20	23	26	42	56
4	20	23	26	42	56
5	20	23	26	42	56
6	20	23	14	42	56
7	20	23	26	42	56
8	20	23	26	42	56
9	20	23	26	40	56
10	20	23	26	42	56

Tabel 5.12 Perhitungan Nilai Similaritas Skenario 3

Percobaan	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	1	0,8	1	1	0,8	1
2	0	0	1	1	1	0,8	1	1	0,8	1
3	0	0	0	1	1	0,8	1	1	0,8	1
4	0	0	0	0	1	0,8	1	1	0,8	1
5	0	0	0	0	0	0,8	1	1	0,8	1
6	0	0	0	0	0	0	0,8	0,8	0,6	0,8
7	0	0	0	0	0	0	0	1	0,8	1
8	0	0	0	0	0	0	0	0	0,8	1
9	0	0	0	0	0	0	0	0	0	0,8
10	0	0	0	0	0	0	0	0	0	0

Tabel 5.13 Rata-rata Pengujian Nilai Konsistensi

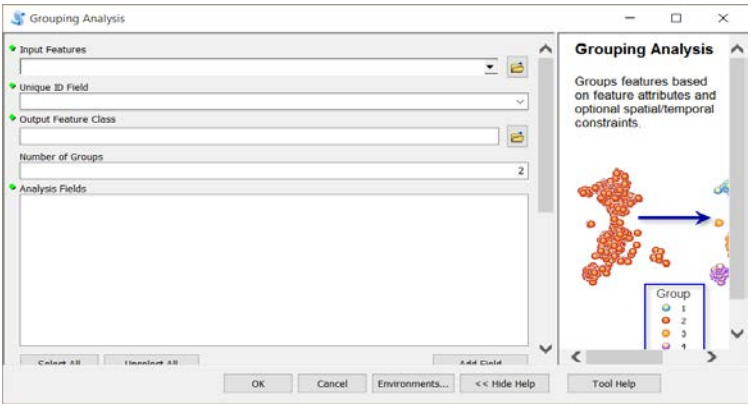
Skenario	Rata-rata Segitiga Atas
1	0,27
2	0,31
3	0,92

Tabel 5.11 menunjukkan hasil percobaan dengan nilai interval = 1000 dan jumlah individu = 200, sedangkan Tabel 5.12 menunjukkan matriks similaritas dengan membandingkan setiap hasil percobaan pada Tabel 5.11. Berdasarkan Tabel 5.13, nilai rata-rata segitiga atas pada setiap skenario secara berturut-turut adalah 0.27, 0.31, dan 0.92. Didapatkan nilai similaritas terbesar adalah 0.92 pada skenario 3 yaitu pada interval = 1000 dan jumlah individu = 200.

5.2.4 Pengujian Evaluasi Hasil

Pada bagian ini, pengujian dilakukan dengan membandingkan hasil pencarian lokasi menggunakan GA dengan hasil *clustering* menggunakan *Grouping Analysis Tools* yang telah tersedia pada ArcGIS 10.3. Proses *clustering* menggunakan *Grouping Analysis Tools* dilakukan dengan menjalankan *tools* pada ArcGIS 10.3 seperti ditunjukkan pada Gambar 5.11. Pada proses ini, yang akan melalui proses *clustering* adalah domain solusi dan menghasilkan domain solusi yang ter-*cluster* seperti pada Gambar 5.12.

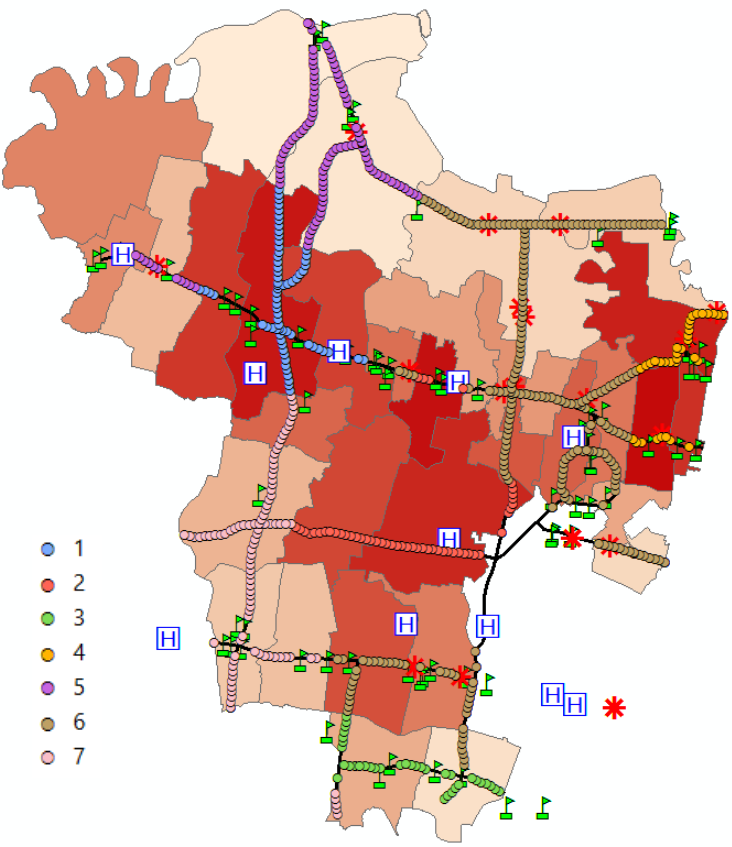
Berdasarkan analisa hasil *clustering* yang ditunjukkan pada Tabel 5.14, *cluster* terbaik secara berturut-turut adalah *cluster* 1, 2, 7, 4, 3, 6, dan 5. Kemudian hasil pencarian lokasi dengan hasil terbaik pada pengujian sebelumnya dibandingkan dengan cara *overlay* hasil pencarian menggunakan GA terhadap hasil *clustering* sebagaimana dilihat pada Gambar 5.13.



Gambar 5.11 Antarmuka *Grouping Analysis tools*

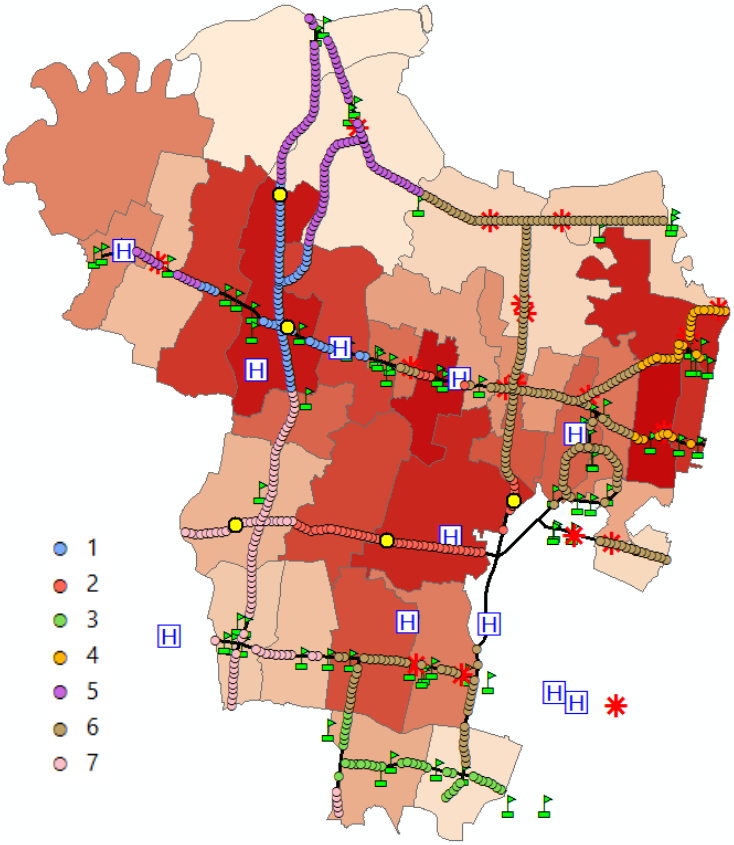
Tabel 5.14 Analisis Hasil *Clustering*

Cluster	Rata - Rata				Total
	Asli		Normalisasi		
	Penduduk	Jarak	Penduduk	Jarak	
1	27172	2249,36	0,8866	0,596136	1,4827
2	25056	2227,30	0,8068	0,58908	1,3959
3	6670	2070,40	0,1139	0,538884	0,6528
4	30181	386,00	1,0000	0	1,0000
5	3648	1111,96	0,0000	0,232252	0,2323
6	9723	686,19	0,2290	0,096037	0,3250
7	8240	3511,72	0,1730	1	1,1730
max	30181	3511,72	1,00	1,00	
min	3648	386,00	0,00	0,00	



Gambar 5.12 Hasil *Clustering*

Berdasarkan Gambar 5.13, pencarian 5 lokasi alternatif SPBU menggunakan GA mendapatkan hasil pencarian di 3 *cluster* terbaik yaitu *cluster* 1, 2, dan 7. Lokasi alternatif hasil pencarian menggunakan GA ditunjukkan dengan lambang titik berwarna kuning.



Gambar 5.13 Overlay GA dengan *Cluster*

5.3 Analisis Hasil Uji Coba

Dari hasil skenario uji coba yang telah dilakukan, beberapa parameter memberikan pengaruh terhadap hasil pencarian alternatif lokasi SPBU. Parameter yang digunakan antara lain adalah nilai jumlah individu sebagai parameter dalam metode GA dan besaran interval yang akan berpengaruh pada jumlah domain solusi. Uji coba dilakukan dengan membandingkan nilai *fitness* dan nilai rasio similaritas.

Dari uji coba hasil pencarian lokasi SPBU terhadap dua jenis masukan jalan, didapatkan bahwa alternatif lokasi hasil pencarian mengarah pada wilayah-wilayah dengan populasi penduduk yang relatif tinggi namun juga tidak mengarah ke daerah yang dekat dengan SPBU eksisting. Hal tersebut dapat ditunjukkan dengan membandingkannya pada hasil analisis *Euclidean Distance* terhadap lokasi SPBU eksisting.

Selanjutnya, hasil uji coba nilai *fitness* dengan berbagai nilai parameter populasi menunjukkan bahwa semakin besar nilai popoulasi maka nilai *fitness* yang didapatkan juga semakin besar. Berdasarkan hasil uji coba, nilai *fitness* yang berada pada *range* maksimal didapatkan ketika *interval* = 500 dengan *jumlah individu* 100 dan 200, serta pada *interval* = 1000 dengan *jumlah individu* 200.

Kemudian pada perhitungan konsistensi, didapatkan nilai similaritas sebesar 0.27 pada skenario pertama dengan jumlah individu = 100, kemudian sebesar 0.31 pada skenario pertama dengan jumlah individu = 200, dan sebesar 0.92 pada skenario kedua dengan jumlah individu = 200. Nilai similaritas maksimum tidak bernilai 1 kemungkinan besar disebabkan oleh pembentukan individu awal pada metode GA yang di-*generate* secara random sehingga memungkinkan *id point* yang sama tidak selalu muncul pada setiap percobaan.

Dan terakhir pada pengujian evaluasi hasil pencarian, ketika dibandingkan dengan hasil *clustering*, hasil pencarian lokasi alternatif SPBU menggunakan GA mendapatkan hasil pencarian di 3 *cluster* terbaik. Hasil pencarian tidak beraglomerasi ke satu cluster dikarenakan pada fungsi *fitness* terdapat fungsi yang menghitung jarak total di dalam satu kombinasi. Hal ini bertujuan untuk mendapatkan alternatif lokasi yang bervariasi.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari proses dan uji coba dari program dan saran untuk pengembangan dari program itu sendiri.

6.1 Kesimpulan

Setelah dilakukan serangkaian uji coba dan analisa, maka dapat diambil kesimpulan sebagai berikut:

1. Penggunaan metode analisis spasial dan Genetic Algorithm dapat melakukan pencarian alternatif lokasi SPBU potensial berdasarkan kriteria yang ditentukan yaitu kriteria jarak dengan SPBU eksisting dan populasi penduduk.
2. Berkenaan dengan metode Genetic Algorithm, perubahan nilai parameter populasi berpengaruh terhadap nilai *fitness* yang diperoleh ketika generasi terakhir tercapai. Pada percobaan yang telah dilakukan, nilai *fitness* terbesar diperoleh ketika nilai jumlah individu = 200 dan interval = 1000.
3. Nilai konsistensi hasil pencarian alternatif lokasi SPBU terbesar adalah sebesar 0.92 nilai jumlah individu = 200 dan interval = 1000 setelah percobaan sebanyak 10 kali terhadap masing-masing skenario parameter yang berbeda. Hal ini dikarenakan pada tahap pembentukan individu awal dilakukan dengan cara random sehingga memungkinkan tidak ter-*cover*-nya seluruh domain solusi untuk masuk ke dalam tahap pencarian.

6.2 Saran

Saran untuk pengembangan selanjutnya dari pengerjaan Tugas akhir ini antara lain:

1. Penambahan kriteria yang lebih spesifik mengenai pemilihan lokasi SPBU yang potensial agar hasil pencarian lokasi SPBU lebih akurat dan optimal.
2. Penggunaan metode pencarian lain sebagai bahan perbandingan dengan *Genetic Algorithm*.
3. Implementasi *web-based tool* karena pada Tugas akhir ini penggunaan *tool* terbatas pada lingkungan perangkat lunak ArcGIS.

DAFTAR PUSTAKA

[1] I. W. K. Giri, "Penentuan Pendirian Lokasi Potensial Stasiun Pengisian Bahan Bakar Umum (SPBU) Se Bandung Raya dengan Menggunakan Geographic Information System (GIS)," Politeknik Pos Indonesia Bandung, Bandung, 2012.

[2] M. A. Haque, "Locating Optimum Location for Well Drilling Using Genetic Algorithms," *IEEE International Conference on Industrial Technology*, 2002.

[3] X. Li and A. G. Yeh, "Integration of Genetic Algorithm and GIS for Optimal Location Search," *International Journal of Geographica Information Science,* vol. 19, no. 5, pp. 581-601, 2005.

[4] University of Maryland Libraries, Introduction to GIS, College Park: McKeldin Library, 2012.

[5] E. Irwansyah, Sistem Informasi Geografis: Prinsip Dasar dan Pengembangan Aplikasi, Yogyakarta: digibooks, 2013.

[6] S. Cholid, Sistem Informasi Geografis: Suatu Pengantar, Bogor: Staff Akademik Departemen Ilmu Kesejahteraan Sosial FISIP UI, 2009.

[7] Y. Sadahiro, Course #716-26 Advanced Urban Analysis E. Lecture Title: – Spatial Analysis using GIS – Associate professor of the Department of Urban, Japan: Engineering, University of Tokyo, 2006.

[8] ESRI, ARCGIS 9, Redlands: ESRI, 2004.

[9] T. Rauch, J. Lohmeier and C. Wyley, Handbook on Development Design - Preface, Berlin: Baobab, 1997.

[10] E. B. Santoso, E. Umilia and B. U. Aulia, Diktat Analisis Lokasi dan Keruangan, Surabaya: Program Studi Perencanaan Wilayah dan Kota, Fakultas Teknik Sipil dan Perencanaan, Institut Teknologi Sepuluh Nopember, 2012.

[11] J. F. Engel, R. D. Blackwell and P. W. Miniard, Consumer Behaviour, Texas: Dryden Press, 1995.

[12] A. Widhiyasa, "Microsoft Word - KAJIAN GENETIC ALGORITHM.doc," 2007. [Online]. Available: http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2006-2007/Makalah/Makalah0607-119.pdf. [Accessed 17 December 2015].

[13] X.-B. Hu and E. Di Paolo, "An efficient genetic algorithm with uniform crossover for air traffic control," *Computers and Operations Research*, vol. 36, no. 1, pp. 245-259, 2009.

BIODATA PENULIS

Muhammad Yarjuna Rohmat atau biasa dipanggil Rohmat, dilahirkan di Bandung 1 Februari 1995. Merupakan anak kedua dari empat bersaudara. Sampai dengan saat ini penulis telah menempuh pendidikan formal di MI Zakaria, SMP Terpadu Baiturrahman, SMA Terpadu Baiturrahman. Setelah lulus SMA penulis melanjutkan pendidikan ke jenjang perkuliahan di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Bidang studi yang diambil oleh penulis pada saat kuliah di Teknik Informatika ITS adalah Manajemen Informasi.

Selama menempuh perkuliahan penulis aktif sebagai anggota unit kegiatan mahasiswa bola basket. Penulis juga aktif dalam kegiatan kepanitian Schematics sebagai relawan pada Reeva 2013 dan 2014. Selain itu penulis juga aktif dalam kegiatan komunitas jurusan yaitu Pecinta Alam Mahasiswa Informatika (PAMOR) ITS sebagai anggota pada tahun 2013-2014, staf divisi hubungan masyarakat pada tahun 2014-2015, dan ketua divisi kegiatan pada tahun 2015-2016.